Titre: Title:	Un algorithme de génération de colonnes pour le problème de tournées de véhicule avec demandes stochastiques
Auteur: Author:	Charles Gauvin
Date:	2012
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Gauvin, C. (2012). Un algorithme de génération de colonnes pour le problème de tournées de véhicule avec demandes stochastiques [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1018/

UNIVERSITÉ DE MONTRÉAL

UN ALGORITHME DE GÉNÉRATION DE COLONNES POUR LE PROBLÈME DE TOURNÉES DE VÉHICULE AVEC DEMANDES STOCHASTIQUES

CHARLES GAUVIN
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2012

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

UN ALGORITHME DE GÉNÉRATION DE COLONNES POUR LE PROBLÈME DE TOURNÉES DE VÉHICULE AVEC DEMANDES STOCHASTIQUES

présenté par : GAUVIN Charles
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

M. ROUSSEAU Louis-Martin, Ph. D., président
M. DESAULNIERS Guy, Ph. D., membre et directeur de recherche
M. GENDREAU Michel, Ph. D., membre et codirecteur de recherche
M. REI Walter, Ph. D., membre

À tous mes collègues du GERAD,
vous me manquerez...

REMERCIEMENTS

Je tiens tout d'abord à remercier Hocine Bouarab, sans qui ce mémoire et cette maîtrise aurait été beaucoup plus difficile à réaliser. Ses connaissances approfondies en recherche opérationnelle et ses habiletés mathématiques se sont révélés essentielles dès les débuts de mes études à l'école Polytechnique. C'est toutefois son amitié et son expérience personnelle qui m'ont été le plus importantes.

Je tiens également à remercier mes deux codirecteurs, Guy Desaulniers et Michel Gendreau pour leur soutien. Je suis particulièrement reconnaissant de l'encadrement et de l'aide que Mr. Desaulniers m'a accordé, et ce malgré ses nombreuses responsabilités et obligations.

Le support de François Lessard s'est également avéré d'une valeur inestimable. Sa compétence, son efficacité ainsi que sa serviabilité ont contribué à rendre le travail de programmation beaucoup plus agréable et enrichissant.

Finalement, je tiens à remercier tous mes professeurs et collègues du GERAD. Votre intérêt et votre travail ont été une très grande source de motivation au cours de ces deux années et demi.

RÉSUMÉ

Ce mémoire présente un algorithme exact de génération de colonnes avec plans coupants pour le problème de tournées de véhicules avec demandes stochastiques. Nous posons le problème comme un programme stochastique en deux étapes en nombres entiers et adoptons une formulation se basant sur le graphe espace-état associé. Nous utilisons ensuite la décomposition de Dantzig-Wolfe pour produire un problème maître de partitionnement d'ensemble et un sous-problème de plus court chemin avec contraintes de ressources.

Nous proposons de résoudre ce modèle à l'aide d'un algorithme exploitant des techniques à la fine pointe de l'état de l'art. Lors de la résolution du sous-problème, nous ne nous limitons pas uniquement aux routes sans 2-cycles ; nous effectuons également des expériences avec les ng-routes ainsi qu'avec les chemins élémentaires. Pour implémenter ces concepts et éliminer de plus grands cycles, nous ajoutons des ressources de visite binaires indiquant si un client peut encore être visité dans une prolongation de la route élémentaire courante.

Afin de résoudre le sous-problème, nous utilisons un algorithme d'étiquetage bidirectionnel tirant parti de l'acyclicité du graphe espace-état et qui considère uniquement une fois chaque nœud. Dans le but de limiter le nombre d'étiquettes générées à chaque itération de cette procédure, nous introduisons une règle de dominance améliorée qui exploite la structure du graphe sous-jacent. Nous utilisons également le concept de clients non atteignables pour favoriser l'élimination d'étiquettes inutiles. Par ailleurs, nous implantons une méthode de recherche taboue qui accélère l'obtention de chemins réalisables de coût réduit négatif.

Pour augmenter la borne inférieure trouvée à chaque itération de l'algorithme de génération de colonnes, nous ajoutons deux types d'inégalités valides au problème maître : des contraintes de capacité et des contraintes de sous-ensemble de lignes. Nous modifions la structure du sous-problème afin de prendre en compte ces coupes que nous générons dynamiquement à chaque nœud de branchement de façon heuristique. Si l'algorithme de séparation ne parvient pas à trouver d'inégalités valides suffisamment violées par la solution fractionnaire actuelle, alors nous procédons à un branchement basé sur les inter-tâches.

Finalement, nous présentons des résultats numériques qui démontrent la compétitivité de notre algorithme. Notre méthode permet effectivement de résoudre 20 nouvelles instances tirées de la littérature en moins de 20 minutes en plus d'accélérer considérablement la réso-

lution des instances déjà résolues. Seulement 2 des 40 instances de notre ensemble de tests demeurent irrésolues.

ABSTRACT

This master's thesis presents an exact branch-cut-and-price algorithm for the vehicle Routing problem with stochastic demands. We formulate the problem as a two stage integer stochastic program with fixed recourse and adopt a formulation based on the associated state-space graph. We explain how this model can be transformed into a set partitioning master problem and an associated shortest path problem with resource constraint subproblem using Dantzig-Wolfe decomposition.

We use a column generation algorithm based on state of the art techniques. During the resolution of the subproblem, we do not limit ourselves to the generation of routes without 2-cycles. We also experiment with *ng*-routes as well as elementary routes. In order to implement these concepts and eliminate larger cycles, we introduce additional binary resource variables each indicating wether a client can be visited by an extension of the current route.

We use a bidirectional label setting algorithm that exploits the acyclicity of the underlying graph and only needs to consider each vertex exactly once. To limit the number of labels that can be generated at any iteration of that algorithm, we introduce an improved dominance rule that exploits the structure of the space-capacity graph. We also promote elimination of dominated labels by using the concept of unreachable vertices. In addition, we utilize a tabu search heuristic to speed up the identification of feasible negative reduced cost routes.

To improve the lower bound found at each iteration of the column generation procedure, we introduce valid inequalities in the master problem. We specifically consider capacity cuts and Subset-Row Inequalities. We identify violated cuts dynamically at each node of the Branch-and-Bound tree by using a heuristic procedure. If the separation algorithm fails to identify violated cuts, we proceed to normal branching based on inter-tasks. The dominance rule and structure of the subproblem are modified to take these new inequalities into account.

Finally, we present numerical results that prove the competitiveness of our algorithm. Indeed, we manage to solve to optimality 20 new instances taken from the literature in less than 20 minutes and we considerably improve the computational time for those already closed. Only 2 out of the 40 instances of our test set remain unsolved.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

Abbrévation	Signification en anglais	Signification en français
ACC	Aggregate capacity cut	Coupe de capacité agrégée
B&B	Branch-and-bound	Énumération implicite
BC	Branch-and-cut	Énumération implicite avec plans coupants
BCP	Branch-cut-and-price	Énumération implicite avec génération de colonnes et plans coupants
BP	Branch-and-price	Énumération implicite avec génération de colonnes
CC	Capacity cut	Coupe de capacité
DEP	Deterministic equivalent problem	Problème équivalent déterministe
ESPPRC	Elementary shortest path problem with resource constraints	Problème de plus court chemin élémentaire avec contraintes de ressources
EEV	Expected result of using expected value solution	Résultat en utilisant la solution avec valeur moyenne
EV	Expected value solution	Solution avec valeur moyenne
MIP	Mixed integer program	Programme en nombres entiers mixte
PM	Master Problem (MP)	Problème maître
PMR	Restricted master problem (RMP)	Problème maître restreint
RP	Recourse problem	Problème avec recours
SP	Subproblem	Sous-problème
SPP	Shortest path problem	Problème de plus court chemin
SPPRC	Shortest path problem with resource constraints	Problème de plus court chemin avec contraintes de ressources
SPPRC-κ-cyc	Shortest path problem with resource constraints and κ-cycle elimination	Problème de plus court chemin avec contraintes de ressources et élimination de κ-cycles
SRI	Subset-row inequalities	Inégalités de sous-ensemble de lignes
TSP	Travelling salesman problem	Problème du voyageur de commerce
VRP	Vehicle routing problem	Problème de tournées de véhicules
VRPSD	Vehicle routing problem with stochastic demands	Problème de tournées de véhicules avec demandes stochastiques
VRPTW	Vehicule routing problem with time windows	Problème de tournées de véhicules avec fenêtres de temps
VSS	Value of stochastic solution	Valeur de la solution stochastique

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

1.1 Définitions et concepts de base

Les problèmes de tournées de véhicules constituent une classe de problèmes majeure en recherche opérationnelle. Ces problèmes relèvent d'une grande importance pratique pour de nombreuses industries, car le transport de marchandises représente une partie importante du coût total d'un produit. Ce type de problème est également primordial pour les compagnies se spécialisant dans la livraison ou la collecte de biens ou de personnes comme UPS, FedEx et Postes Canada.

Le problème de tournées de véhicules de base (« vehicle routing problem » - VRP) consiste à déterminer un ensemble de routes couvrant à coût minimum un ensemble de clients ayant des demandes connues à l'aide d'un flotte de véhicules identiques de capacité finie basés à un dépôt central. Pour refléter les spécificités et contraintes des problèmes réels, plusieurs variantes de ce problème ont également été formulées. Parmi les généralisations du VRP les plus étudiées, on peut citer le problème de tournées de véhicules avec fenêtres de temps (« vehicle routing problem with time windows » - VRPTW). Ce problème cherche comme le VRP à déterminer un ensemble de routes optimales pour une flotte de véhicules de capacité finie, mais dans ce cas, les livraisons ou les collectes doivent avoir lieu dans un intervalle de temps préétabli.

Récemment, plusieurs efforts ont été effectués pour créer des problèmes de tournées de véhicule « riches » incorporant des contraintes plus complexes ainsi que des éléments aléatoires pour augmenter la robustesse des modèles. Ce mémoire s'inscrit en partie dans ce thème de recherche. Il porte sur le problème de tournées de véhicules avec demandes stochastiques (« Vehicle routing problem with stochastic demands » - VRPSD) et peut s'énoncer comme suit. Étant donné un ensemble de clients avec demandes aléatoires et une flotte de véhicules identiques possédant une capacité finie basés à un dépôt unique, quel est l'ensemble des routes partant et se terminant au dépôt qui permet de visiter tous les clients exactement une fois tout en minimisant la somme totale espérée des distances parcourues ?

Contrairement au VRP, les demandes des clients sont stochastiques dans le VRPSD. Il

est donc impossible de savoir précisément comment assigner les véhicules afin de satisfaire toutes les demandes avant la révélation de la valeur de ces variables aléatoires. Cependant, il est raisonnable de supposer qu'on doit planifier l'ensemble des tournées de façon préalable avant d'avoir obtenu toute l'information sur les demandes des clients. Par exemple, il est possible que les horaires des chauffeurs doivent être établis quelques jours à l'avance et que ces derniers ne peuvent pas être appelés au besoin.

Afin de contourner cette difficulté, on suppose que l'on confectionne dans un premier temps un ensemble de routes visitant tous les clients. Cette affectation se fait à priori, c.-à-d. avant de connaître la valeur des variables aléatoires. Puis, lorsque vient le temps de livrer ou de collecter les biens, les véhicules suivent les itinéraires préétablis. Cette méthodologie a comme avantage potentiel de créer des tournées stables pour les chauffeurs et les clients et semble applicable dans plusieurs contextes réels.

Comme les demandes des clients sont des variables aléatoires, il est possible que la demande cumulée réalisée d'une route fixée préalablement dépasse la capacité d'un véhicule. On dit alors qu'un échec se produit et on applique une stratégie de *recours*. La stratégie que l'on utilise dans notre cas consiste à épuiser la capacité résiduelle du véhicule au client où l'échec se produit et à retourner au dépôt pour se ravitailler avant de continuer la route préétablie. Ceci correspond au seul cas où l'on permet une livraison partielle à un client.

Le coût d'un échec au client i est donc donné par la distance d'un aller-retour de i au dépôt. Comme les distances sont supposées symétriques, ceci correspond à deux fois la distance de i au dépôt. Si la demande du client i est exactement égale à la capacité résiduelle du véhicule, alors on suppose que l'on continue au prochain client $i+1$. Si $i+1$ n'est pas le dépôt, alors l'échec a lieu au client $i+1$; sinon on termine la route normalement. Cette stratégie de mise à jour de la route se nomme *recours*.

Lorsque la demande totale d'une route est suffisamment grande, il est possible qu'il y ait plus d'un échec sur cette route. On dit alors que le u^e échec se produit exactement au client i si la demande totale satisfaite par le véhicule jusqu'au client i est plus grande à u fois la capacité et que la demande totale satisfaite par le véhicule jusqu'au client précédent $i-1$ ne dépasse pas u fois la capacité.

Le problème cherche donc à déterminer l'ensemble des routes qui minimisent la somme totale des coûts de déplacement ; c'est-à-dire la somme des distances parcourues le long des

routes préétablies et le coût espéré d'échec à chaque client étant donné ces routes.

1.2 Optique du mémoire

Bien que le VRPSD ait de nombreuses applications concrètes, ce mémoire se place dans un cadre académique. Ainsi, les données sur les demandes des clients, les distances entre ceux-ci ainsi que la capacité des véhicules sont tirées des instances standard de la littérature. Nous faisons aussi plusieurs hypothèses sur la distribution des demandes des clients. Nous supposons entre autres que l'espérance des demandes est inférieure à la capacité d'un véhicule. Cette prémisse est raisonnable, car il serait peu probable qu'une compagnie d'offrir ses services à un client si elle ne peut satisfaire sa demande. Il serait possible de servir un client avec une demande excédant la capacité d'un véhicule s'il était possible d'effectuer des livraisons ou des cueillettes partielles. Nous supposons néanmoins que les bien livrés ou cueillis ne peuvent être divisés.

Comme plusieurs chercheurs, nous faisons également l'hypothèse plus forte que les routes confectionnées sont réalisables en moyenne. Il est normal d'utiliser ce postulat car, dans le cas adverse, il existerait des routes systématiquement sur et sous utilisées. En outre, il ne serait pas logique d'un point de vue stratégique et décisionnel d'utiliser une flotte de véhicules avec des capacités insuffisantes en moyenne pour couvrir une route. Plutôt que de concevoir des tournées étant presque certaines d'échouer, il serait beaucoup plus efficace et logique de considérer le choix de plusieurs routes réalisables en moyenne.

En outre, nous supposons que les demandes sont indépendantes et identiquement distribuées et qu'elles sont connues uniquement au moment d'arriver à un client donné. Ceci semble raisonnable d'un point de vue réel. Nous considérons également que chacune des demandes suit une distribution de Poisson. Cette dernière supposition n'est pas nécessairement appuyée par des évidences empiriques. Cependant, la distribution de Poisson possède des caractéristiques fidèles à de nombreuses situations réelles comme le fait d'être définie sur les entiers non négatifs. Ce type de distribution est également couramment utilisé dans la littérature, il est donc plus facile de comparer nos résultats. Par ailleurs, comme l'espérance et la variance sont égales, elle permet des simplifications qui réduisent la complexité du problème et possède des propriétés importantes comme l'additivité.

Finalement, nous supposons que l'espérance des demandes est entière. Bien qu'elle puisse

sembler discutable, cette seconde supposition est acceptable dans le cas déterministe. En effet, notre méthode de résolution permet toujours de discrétiser les demandes cumulées espérées selon la précision désirée. Toutefois, comme on le mentionne à la section 3.3.2, cette discrétisation ne peut plus être appliquée au cas stochastique, car le support de la distribution utilisée est l'ensemble des nombres entiers naturels. On doit cependant faire cette hypothèse, car elle est essentielle pour la résolution de notre modèle. Dans tous les cas, les résultats de la section 5 indiquent qu'il est préférable d'utiliser des demandes stochastiques d'espérance entière plutôt que d'ignorer la variation aléatoire de celles-ci.

1.3 Objectifs de recherche

L'objectif de ce mémoire est de déterminer si la méthode de génération de colonnes avec plans coupants (« Branch-cut-and-price » - BCP) peut être utilisée efficacement pour résoudre le VRPSD lorsque ce dernier est formulé comme un programme stochastique en deux étapes en nombres entiers. Nous tenterons spécifiquement d'utiliser la formulation basée sur la décomposition de Dantzig-Wolfe proposée par (Christiansen et Lysgaard, 2007). Nous utiliserons également l'algorithme de base de génération de colonnes (« branch-and-price » - BP) conçu par ces derniers et tenterons de l'améliorer à l'aide du cadre de résolution proposé par (Desaulniers *et al.*, 2008) et de certains concepts tirés de (Baldacci *et al.*, 2010). Nous cherchons à accélérer la résolution de ce problème par rapport à l'algorithme de BP de (Christiansen et Lysgaard, 2007) et de résoudre des instances non résolues jusque là.

1.4 Plan du mémoire

La section 2 présente tout d'abord une revue de littérature des travaux effectués sur les différents types de problèmes de tournées de véhicules stochastiques. Elle se penche particulièrement sur les articles traitant du VRPSD. Nous présentons également plusieurs travaux pertinents portant sur les méthodes de résolution à la fine pointe de l'état de l'art en termes de génération de colonnes.

La section 3 place ensuite le VRPSD dans un cadre de programmation stochastique à deux étapes avec recours. Nous décrivons ensuite sa formulation mathématique et expliquons pourquoi celle-ci se prête bien à la décomposition de Dantzig-Wolfe qui consiste à décomposer le problème original en un problème maître et un ou plusieurs sous-problèmes.

Nous nous penchons ensuite en détail sur l'algorithme de BCP utilisé pour résoudre notre

modèle dans la section 4. Nous présentons le cadre de résolution générique et on étudie ensuite les différentes modifications pouvant être appliquées. Nous considérons particulièrement les améliorations apportées au sous-problème ainsi que les différentes règles de branchement et inégalités valides pouvant être ajoutées pour renforcer la formulation et obtenir des solutions entières.

La section 5 présente des résultats numériques obtenus à la suite de plusieurs expériences effectuées avec différentes stratégies de résolution. Nous comparons ensuite l'impact des valeurs de certains paramètres sur plusieurs métriques telles que le temps de résolution et nous évaluons l'interaction entre ces paramètres.

Finalement, la section 6 résume les contributions de ce mémoire. Nous mentionnons également certaines limitations de notre travail ainsi que des travaux futurs potentiels.

CHAPITRE 2

REVUE DE LITTÉRATURE

Tel que décrit dans (Gendreau *et al.*, 1996a), les problèmes de tournées de véhicules peuvent comporter plusieurs paramètres aléatoires dont on ignore précisément les valeurs à un certain moment du processus décisionnel. La littérature scientifique sur les VRP stochastiques (« stochastic vehicle routing problems » - SVRP) s'est toutefois surtout limitée à considérer comme stochastique les demandes et la présence de clients, les temps de déplacement et de service, ainsi que les fenêtres de temps.

Les différents SVRP peuvent être modélisés de plusieurs façons, notamment comme des processus de décision markovien ou des programmes stochastiques à plusieurs étapes. Les algorithmes exacts appartiennent toutefois en quasi-totalité à la deuxième classe de modèles. Plus spécifiquement, les programmes stochastiques pour le SVRP se décomposent en deux grande familles : les programmes avec contraintes probabilistes et les programmes avec recours. Ces programmes sont généralement formulés comme des programmes stochastiques en deux étapes.

Les programmes avec contraintes probabilistes pour le SVRP cherchent à obtenir des solutions réalisables de coût minimal où la probabilité de respecter certaines contraintes ayant des coefficients aléatoires est supérieur à un seuil fixé. Par exemple, (Laporte *et al.*, 1992) proposent un programme stochastique en deux étapes avec contraintes probabilistes pour le problème de tournées de véhicules sans capacité avec temps de déplacement stochastiques. Leur modèle impose que le probabilité que le temps de déplacement total d'une route dépasse un certain seuil soit inférieur à une probabilité donnée.

(Jula *et al.*, 2006) étudient un VRP avec fenêtres de temps dures où l'on permet l'attente à un client si un véhicule arrive avant le début de la fenêtre de temps associée. Leur modèle considère également des temps de déplacement et service stochastiques et un seul véhicule sans contraintes de capacité. Ces auteurs incorporent également des contraintes probabilistes imposant le choix de routes ayant une probabilité de satisfaire toutes les fenêtres suffisamment grande. Ce modèle est ensuite résolu à l'aide d'une heuristique basée sur le programmation dynamique.

À l'opposé, les programmes stochastiques avec recours pour le SVRP cherchent à obtenir une solution préliminaire pouvant être irréalisable, mais que l'on corrige ou qu'on améliore grâce à une stratégie de recours. Ces programmes tentent habituellement de minimiser l'espérance du coût de déplacement total sujet à certaines contraintes. Ce coût de déplacement espéré est souvent décomposable en une composante déterministe (*e.g.* le coût de déplacement déterministe) et une partie stochastique (le coût de recours). Certains modèles incorporent toutefois des coûts déterministes additionnels comme un coût fixe associé à chaque véhicule ainsi que des coûts stochastiques déterminés par une stratégie de recours complexe. Dans la plupart des cas, les variables de décisions de première étape constituent le choix de routes établies à priori ainsi que le nombre de véhicules assignés Les variables de seconde étape constituent quant à elle les corrections ou pénalités apportées aux routes initiales.

Il est cependant possible d'utiliser un nombre arbitrairement grand de variables de décisions différentes et de modifier l'ordre dans lequel on prend ces décisions. De plus, il peut exister un nombre arbitrairement grand de stratégies de correction, selon les politiques réelles des compagnies, le moment auquel les valeurs des variables aléatoires sont révélées ainsi que la possibilité de résolution ou de réoptimisation des problèmes en un temps donné. Les travaux académiques se sont cependant penchés sur des recours relativement simples et facilement calculables dans un contexte de programmation en 2 étapes.

À titre d'exemple, (Laporte *et al.*, 1992) étudient également 2 modèles avec recours où les routes dont le temps de déplacement total dépassent un seuil fixé sont pénalisées. Comme on ne fait que pénaliser les routes irréalisables par rapport à l'excès de temps, ce modèle correspond à un recours simple et il est facile de déterminer une borne inférieure sur le coût espéré optimal. Afin de résoudre ce modèle, ces auteurs utilisent une adaptation de la méthode L-shaped en nombres entiers.

En comparaison, (Spliet et Gabor, 2012) utilisent une stratégie de recours extrêmement complexe où l'ordre des décisions ne correspond pas à celui des SVRP standards. En effet, ces auteurs ont considéré un programme stochastique en deux étapes avec recours pour le problème de tournées de véhicules avec assignation de fenêtres de temps et demandes stochastiques (« Time window assignment vehicle routing problem »- TWAVRP). La première étape consiste d'abord à assigner des fenêtres de temps de service à des clients avant de connaître précisément leur demande de telles sorte que l'on puisse toujours concevoir des routes réalisables par rapport à ces fenêtres dans tous les scénarios de demandes possibles. Lorsque ces demandes sont révélées, on cherche ensuite à confectionner des routes dont le

coût de déplacement total est minimisé. La stratégie de recours consiste donc à résoudre un VRPTW et de déterminer un ensemble de routes minimales respectant les contraintes de temps imposées dans la première étape. L'objectif du problème global est de minimiser le coût de déplacement espéré selon l'ensemble des scénarios de demande réalisables.

Bien qu'intéressants, ces travaux ne sont pas directement applicables à notre problème, car nous n'utilisons pas de fenêtres de temps ni de temps de service et considérons uniquement la demande des clients comme stochastiques, c.-à-d. le VRPSD. Le VRPSD est toutefois le problème qui a reçu le plus d'attention de la part des chercheurs parmi les variantes stochastiques du VRP.

(Tillman, 1969) fut le premier à étudier ce problème. Il proposa un modèle avec dépôt multiples et demandes aléatoires d'espérance identiques suivant une distribution de Poisson. Son modèle suppose également qu'il existe des pénalités pour terminer une route sans avoir utilisé toute la capacité du véhicule ou de la terminer sans avoir satisfait la demande de tous les clients. Pour résoudre ce problème, il utilisa une méthode en deux étapes séquentielles. La première partie de sa méthode est une adaptation de l'heuristique de (Clarke et Wright, 1964) basée sur le calcul des épargnes faites en assignant un client à une route réalisable donnée. Dans la deuxième partie de l'algorithme, des véhicules de capacité différentes sont ensuite assignés à la nouvelle route afin de minimiser l'espérance du coût de sur/sous-utilisation.

Un autre article important est attribuable à (Golden et Stewart, 1983). Ces derniers ont considéré un VRP avec capacité fixe et demandes aléatoires où un échec survient lorsque la charge d'un véhicule dépasse sa capacité sur une route donnée. Ils ont proposé un modèle avec contraintes probabilistes ainsi que deux modèles avec recours. Le premier modèle avec recours assigne un coût proportionnel à la probabilité d'obtenir un échec alors que le deuxième assigne une pénalité proportionnelle à l'échec survenu. Ces chercheurs ont ensuite résolu leur modèle grâce à deux heuristiques. Ils ont également effectué plusieurs expériences à l'aide de distributions de demandes différentes.

(Bertsimas, 1992) a réutilisé le concept d'optimisation a priori utilisé par (Jaillet, 1985) pour le problème de voyageur de commerce probabiliste (« Probabilistic travelling salesman problem ») et l'a appliqué au VRPSD. Ce dernier a notamment établi deux stratégies de recours dans le cas où des routes étaient fixées préalablement. L'une suppose que les demandes des clients sont connues suffisamment tôt dans le processus décisionnel pour réoptimiser les routes alors que l'autre suppose que l'on peut uniquement subir les décisions prises lors de

la première étape. Dans les deux cas, on dit qu'un échec survient lorsque la capacité résiduelle du véhicule est insuffisante pour couvrir un client. Quand cette situation survient, on retourne au dépôt et on visite le prochain client. Ses travaux ont démontré que l'approche a priori était non seulement réaliste d'un point de vue pratique, mais menait également à des solutions comparables à celles obtenues par la stratégie de réoptimisation complète en moyenne.

(Laporte et Louveaux, 1993) ont modélisé le VRPSD à l'aide d'une formulation de flot sur les arcs à deux indices indiquant combien de véhicules empruntent un arc donné dans une solution. Cette formulation est dite compacte ou non décomposée par opposition à la formulation de Dantzig-Wolfe qui se base sur un problème maître et un sous-problème. Ces auteurs ont ensuite étudié les propriétés de cette formulation et ont observé qu'il était très difficile de résoudre des instances non-triviales exactement avec une stratégie de recours générale et sans supposer que les routes initiales ne dépassaient pas la capacité d'un véhicule. Ils ont donc utilisé un recours consistant à retourner au dépôt en cas d'échec et ont supposé que les routes confectionnées à priori était réalisables. Dans ce cas, la stratégie de recours représente un forme particulière de la classe générique des recours fixes et complets : le recours simple. Afin de prendre en compte cette stratégie de recours, ils suffit de modifier la fonction objectif de base du VRP en lui ajoutant une fonction de pénalité pour calculer le coût espéré d'échec. Ces chercheurs ont généralisé ces résultats et ont proposé un modèle général pour la résolution de programmes stochastiques en deux étapes et en nombres entiers avec recours complet.

Ces auteurs ont démontré que l'utilisation de la méthode L-shaped en nombres entiers permettait de résoudre ce modèle exact efficacement. Plus spécifiquement, leur procédure se base sur l'identification dynamique de coupes de capacité/sous-tours et l'introduction de règles de branchement afin d'obtenir une solution entière et réalisable. Lorsqu'une solution entière est obtenue, des coupes d'optimalité sont ajoutées si le coût total espéré de la nouvelle solution est inférieur au meilleur coût obtenu jusque là. Des bornes inférieures sur le coût réel de la solution optimales peuvent également être ajoutées avant d'obtenir une solution entière.

Les travaux de (Gendreau *et al.*, 1995) s'inscrivent dans ce cadre de résolution exact. Ils utilisent l'algorithme L-shaped en nombres entiers ainsi qu'une borne inférieure sur le coût d'échec afin d'accélérer l'obtention de coupes d'optimalité. Leur modèle est toutefois plus riche que celui de (Laporte et Louveaux, 1993), car ils considèrent deux niveaux d'incertitude : la variation aléatoire de la demande des clients ainsi que la probabilité de présence de ces derniers. (Hjorring et Holt, 1999) ont réutilisé plusieurs concepts de ces deux derniers travaux. Ils ont également accéléré la résolution du problème grâce à l'identification de coupes

d'optimalité plus efficaces. Leur problème considère toutefois uniquement le cas avec un seul véhicule et demandes stochastiques. Bien que ces travaux relèvent d'une importance majeure, ils ont uniquement permis de résoudre des instances de taille modeste où la demande totale espérée est faible par rapport à la capacité des véhicules et où peu de véhicules sont nécessaires.

(Gendreau *et al.*, 1996b) ont tenté d'éviter l'explosion du temps de calcul engendrée par l'utilisation de ces méthodes exactes pour la résolution de plus grandes instances en utilisant une heuristique. Ces auteurs ont donc implémenté un algorithme de recherche tabou *TabuStoch* permettant d'obtenir des solutions entières rapidement. Comme le calcul du coût espéré d'échec étant donné une solution de première étape nécessite un temps considérable, ces derniers ont utilisé une borne inférieure rapidement calculable sur le coût de recours. Les résultats obtenus se sont révélés très près des solutions optimales pour les instances solubles avec une méthode exacte.

(Laporte *et al.*, 2002) se sont repenchés vers les méthodes exactes et ont amélioré les résultats de (Hjorring et Holt, 1999), (Laporte et Louveaux, 1993) et (Gendreau *et al.*, 1995) en réutilisant plusieurs concepts de ces articles. Ces chercheurs ont rapporté des résultats prometteurs. Ils sont notamment parvenus à résoudre des problèmes contenant jusqu'à 100 clients avec deux véhicules et ont réussi à résoudre des instances avec 4 véhicules et 25 clients. Néanmoins, leur méthode a uniquement réussi à résoudre de grandes instances lorsque le nombre de véhicules requis était faible.

Il semble logique que la complexité des instance augmente rapidement selon le nombre de véhicules nécessaires, car l'espace des solutions augmente considérablement. Il faut considérer plusieurs routes possibles et de nombreuses inégalités de sous-tour doivent être ajoutées avant d'obtenir une solution entière. Lorsque qu'un solution entière est obtenue, des coupes d'optimalité sont ajoutées. Plusieurs d'entre elles doivent être ajoutées avant d'obtenir la solution optimale.

On note également que la difficulté des instances augmente à mesure que la somme espérée des demandes augmente par rapport à la capacité des véhicules multipliée par le nombre de véhicules. Lorsque ce ratio - le taux de remplissage espéré - est élevé, la contrainte de capacité est restrictive par rapport à une route entière, mais faible par rapport aux demandes individuelles. Dans ce cas, les routes sont très chargées et la probabilité d'obtenir des routes sans échec est très faible. On doit donc ajouter plusieurs coupes d'optimalité et bornes inférieures sur le coût de la vraie solution avant d'obtenir le coût optimal. La complexité est encore pire

si on augmente aussi le nombre de clients, car dans ce cas il existe un grand nombre de routes réalisables et plusieurs combinaisons doivent être testées avant de bien borner le coût de la solution réelle.

Afin de résoudre exactement de plus grandes instances lorsqu'il existe peu de routes réalisables et que le nombre de véhicules nécessaires est grand, (Christiansen et Lysgaard, 2007) ont proposé une méthode de génération de colonnes (BP). Cette méthode utilise un formulation basée sur la décomposition de Dantzig-Wolfe qui diffère de la formulation compacte utilisée par la plupart des travaux précédents. Ces derniers ont justifié ce choix en affirmant que la génération de colonnes se révélait particulièrement utile pour les problèmes comme le VRPTW lorsque les fenêtres de temps étaient serrées. Dans ce cas, le domaine réalisable du sous-problème est réduit et il existe moins de routes réalisables à générer ; ce qui accélère la résolution du problème.

Ces chercheurs ont exploité le fait qu'il suffit de connaître la séquence de visite des clients dans une route afin de déterminer son coût espéré d'échec. À l'aide de cet ordre, on sait exactement quelle demande espérée cumulée a été livrée lorsqu'un véhicule arrive à un client donné. En utilisant un graphe espace-état qui indique précisément quelle demande cumulée est livrée jusqu'à un client donné, on peut donc inclure directement le coût espéré d'échec sur chacun des arcs de ce graphe auxiliaire. Contrairement à la méthode L-shaped, leur approche permet de déterminer le coût d'échec d'une route sans connaître préalablement de solution entière réalisable. Il n'est donc pas nécessaire d'approximer la fonction de coût par une borne inférieure.

Le coût espéré d'échec associé à chaque chemin est bien défini et peut être décomposé selon les arcs visités dans le graphe espace-état. Ils ont donc pu utiliser un algorithme de plus court chemin avec contraintes de ressources et élimination de 2-cycles (« shortest path problem with resource constraints and 2-cycle elimination » - SPPRC-2-cyc) afin de résoudre le sous-problème et identifier les chemins de coût réduit négatif permettant d'améliorer la solution actuelle.

Comme le graphe espace-état est acyclique, l'élimination de cycles consiste à empêcher le retour à un nœud appartenant à un ensemble de nœuds associés à un client déjà visité. Les routes générées par ce sous-problème peuvent ensuite être introduites dans un problème maître restreint de recouvrement d'ensemble. Des solutions entières sont générées en intégrant la génération de colonnes dans un arbre de branchement lors du processus d'évaluation

et de séparation (« branch and bound » - B&B).

Néanmoins, l'utilisation du graphe espace-état augmente considérablement la taille de l'instance. Même si l'on considère uniquement les routes réalisables en moyenne, on a $O(nQ)$ nœuds et $O(n^2Q)$ arcs comparativement à $O(n)$ nœuds et $O(n^2)$ arcs dans le graphe original avec n clients. C'est ce qui explique en partie la raison pour laquelle ces auteurs ne parviennent à résoudre que peu d'instances avec 40 clients ou plus.

Les avancées récentes dans la résolution des problèmes connexes du VRP et du VRPTW suggèrent que l'algorithme de BP utilisé par (Christiansen et Lysgaard, 2007) pourrait être considérablement amélioré. Leur algorithme serait entre autres accéléré par l'ajout d'inégalités valides ; ce qui donnerait lieu à un algorithme de BCP. Il serait également avantageux de modifier le sous-problème utilisé et d'exploiter des techniques récentes afin d'accélérer le processus de génération de colonnes.

Des progrès importants ont eu lieu récemment dans la résolution du VRPTW. Ces résultats sont notamment attribuables à une meilleure définition du polyèdre de contraintes et l'identification de nombreuses inégalités valides. À ce sujet, on peut citer les travaux importants de (Jepsen *et al.*, 2008). Ces derniers sont parvenus à identifier des familles d'inégalités valides efficaces pour la résolution du VRP et du VRPTW avec génération de colonnes. Ces inégalités, nommées « subset-row inequalities » (SRI) doivent être ajoutées directement au problème maître (PM) et ne peuvent être prises en compte dans la formulation compacte avec variables de flot sur les arcs. Il est toutefois possible de les considérer dans le sous-problème sans trop modifier sa structure.

Les travaux de (Lysgaard *et al.*, 2004) sur le VRP ont également démontré qu'il était possible d'identifier rapidement et efficacement une classe riche d'inégalités valides violées à l'aide d'algorithmes de séparation heuristiques. Lorsqu'ajoutées à la relaxation linéaire de la formulation compacte initiale, ces coupes permettent de renforcer considérablement les bornes inférieures calculées dans l'arbre de branchement.

Les travaux récents de (Lysgaard *et al.*, 2004) et (Jepsen *et al.*, 2008) ont mené à l'identification de familles d'inégalités valides très efficaces pour accélérer l'obtention de solutions entières et la résolution de problèmes de tournées de véhicule. Ces auteurs ont non seulement montré leur validité théorique, mais ils ont également développé des implantations concrètes efficaces permettant d'obtenir de bons résultats numériques. Cependant, ces travaux se basent

sur des algorithmes limités. (Lysgaard *et al.*, 2004) utilisent seulement un algorithme de BC alors que l'algorithme de BCP de (Jepsen *et al.*, 2008) ne considère pas d'inégalités valides pouvant être ajoutées à la formulation initiale du problème.

Or, plusieurs travaux récents démontrent qu'il est très avantageux de combiner les algorithmes de BP et les algorithmes de plans coupants BC ainsi que de considérer des ensembles plus riches d'inégalités valides (Pessoa *et al.*, 2008). Par exemple, (Fukasawa *et al.*, 2006) ont démontré l'utilité d'ajouter des inégalités valides à la génération de colonnes. Ces derniers ont notamment ajouté des coupes de capacité (« capacity cuts » - CC) au problème maître lors de la décomposition de Dantzig-Wolfe. Comme ces inégalités peuvent être exprimées en fonction des variables de flot sur les arcs, ils sont parvenus à décomposer le coût réduit d'une route selon les arcs visités.

L'interaction entre la génération de colonnes et les inégalités valides est particulièrement importante dans le cas du VRPTW. En effet, la relaxation linéaire de la formulation de base de ce problème avec variables de flot sur les arcs se révèle très faible. Tel qu'illustré dans (Cordeau *et al.*, 2002), une solution fractionnaire peut aisément satisfaire les contraintes de capacité et de fenêtres de temps. Les méthodes de résolution exactes les plus performantes pour le VRPTW utilisent donc des techniques pour renforcir cette relaxation. Elles se basent surtout sur des techniques de décomposition mathématique comme la relaxation lagrangienne et la génération de colonnes (Cordeau *et al.*, 2007).

Comme en témoigne la grande quantité de recherche récente sur le sujet, les méthodes BP se sont révélées particulièrement intéressantes pour résoudre ce type de problème (Desaulniers *et al.*, 2010). Les avancées effectuées dans la résolution du sous-problème, les heuristiques permettant l'obtention de meilleurs solutions réalisables ainsi que la génération rapide de colonnes de coût réduit négatif et l'identification d'inégalités valides se sont notamment révélées d'une importance cruciale. On note également que des travaux relativement récents sur les algorithmes de BC ont donné des résultats intéressants. Cette méthode s'appuie toutefois sur l'identification de nombreux types d'inégalités valides et l'utilisation d'heuristiques afin de calculer de bonnes bornes supérieures à chaque nœud de l'arbre de branchement (Bard *et al.*, 2002).

L'algorithme de (Baldacci *et al.*, 2010) représente la technique de résolution exacte la plus efficace à ce jour pour une grande classe de problèmes de tournées de véhicules, incluant le VRP et le VRPTW. Cette méthode combine plusieurs techniques à la fine pointe de l'état

de l'art et permet de résoudre plusieurs types de problèmes de tournées de véhicule ainsi que 167 des 168 instances de base de Solomon pour le VRPTW (Solomon, 1987). Ces auteurs utilisent un algorithme sophistiqué basé sur la génération de colonnes ainsi que trois heuristiques séquentielles permettant d'obtenir rapidement une borne inférieure forte ainsi qu'une heuristique pour obtenir une solution réalisable de qualité. La première heuristique utilise un sous-problème constitué de *ng*-routes. Une *ng*-route constitue une route de l'origine à la destination de coût minimal pouvant contenir des cycles, mais ne visitant pas un ensemble de clients interdits plus d'une fois. Cet ensemble est déterminé de manière dynamique selon le « voisinage » de chaque client fixé au début de l'algorithme. Ce type de routes favorise l'élimination de petits cycles tout en évitant un coût de calcul trop important. Dans les deux premières étapes, ces chercheurs utilisent également une optimisation de sous-gradient afin d'éviter les problèmes communs de la génération de colonnes comme la dégénérescence dans le problème maître. Dans la dernière étape, ils utilisent des routes élémentaires et ajoutent des inégalités valides du type proposé par (Jepsen *et al.*, 2008)

Il est aussi important de mentionner l'algorithme de BCP proposé par (Desaulniers *et al.*, 2008) pour résoudre le VRPTW. Cet algorithme se base sur la génération de colonnes et ajoute des inégalités valides du type k-chemins (k-path cuts) ainsi que des SRI. Afin de bien prendre en compte ces coupes, le sous-problème est modifié et la règle de dominance astucieuse proposée par (Jepsen *et al.*, 2008) est utilisée. Ces auteurs utilisent également un sous-problème amélioré qui ajoute des contraintes d'élémentarité de façon dynamique afin d'éviter l'augmentation importante du temps de calcul engendrée par les contraintes d'élémentarité. Leur algorithme utilise également un algorithme de recherche taboue ainsi que des étiquetages heuristiques qui permettent de générer rapidement des colonnes de coût réduit négatif dans les premières étapes de l'algorithme de génération de colonnes.

CHAPITRE 3

FORMULATION

3.1 Formulation générique dans un cadre de programmation stochastique

La programmation stochastique a été introduite par (Dantzig, 1955). Il s'agit d'un cadre général pour la modélisation de problèmes d'optimisation impliquant des variables aléatoires. Cette technique vise à rendre les modèles d'optimisation plus robustes face à l'incertitude du monde réel tout en conservant une certaine solvabilité.

Lorsqu'il existe des informations incertaines par rapport au futur, on suppose que les décisions peuvent être échelonnées sur différentes étapes du processus décisionnel. Les choix faits à chaque étape sont dynamiques, car ils sont influencés par l'état du système à l'étape actuelle et donc par les décisions prises dans les étapes précédentes. Les choix faits à chaque étape sont également guidés par les informations limitées disponibles à ce moment sur les facteurs aléatoires. Les valeurs réalisées de ces variables aléatoire sont graduellement dévoilées à chacune de ces étapes. On suppose donc que des décisions peuvent être effectuées de manière non anticipative en fonction de données limitées disponibles à un moment précis dans un contexte stochastique.

Comme les décisions prises dans les premières étapes influencent les décisions futures possibles et l'état dans lequel le système se trouve dans les étapes subséquentes, ces décisions relèvent d'une importance cruciale. Il est donc normal que lors des phases préliminaires, on fixe les variables de décision stratégiques ou « importantes ». Dans les étapes subséquentes, on fixe ensuite les variables de décision tactiques ou « secondaires » selon les différents aléas et décisions prises dans les étapes précédentes. Les étapes représentent des phases abstraites de la prise de décision et ne correspondent donc pas nécessairement à une période de temps. Elles dépendant du processus informationnel

Il est possible d'utiliser un horizon de planification arbitrairement grand s'il est possible de réoptimiser le problème à mesure que l'on reçoit des fragments d'information et que cette information arrive séquentiellement à différents moments dans le processus décisionnel. Cette situation peut notamment être modélisée à l'aide d'un arbre de scénarios. Cependant, l'interdépendance entre les différentes étapes de décision implique une augmentation importante

du nombre de variables à optimiser ainsi que du nombre de contraintes lorsqu'on accroit le nombre de phases de décisions. Il est donc courant de se limiter à deux étapes si le problème est de grande taille et s'il se prête à une telle formulation.

La programmation stochastique se décompose en deux grandes catégories : les programmes avec contraintes probabilistes (« chance constrained program ») et les programmes stochastiques avec recours (« stochastic program with recourse »). Ces deux classes de problèmes possèdent leurs limitations et avantages respectifs et peuvent être vues comme complémentaires à certains égards.

Les programmes avec recours sont souvent utilisés lorsqu'il est naturel d'utiliser une mesure corrective pour améliorer ou rectifier l'état dans lequel le problème se situe à l'étape actuellement considérée. Par exemple, il est possible que la réalisation de variables aléatoires mène à une solution ne respectant pas les contraintes du problème. Le recours permet ainsi de pénaliser l'irréalisabilité de cette solution. Il est également possible que le recours cherche à améliorer une solution réalisable étant donné la réalisation de variables aléatoires ainsi que les décisions des étapes précédentes.

Il est possible de considérer plusieurs stratégies de recours très complexes. Ceci confère une très grande flexibilité de modélisation à la programmation avec recours. Cependant, la résolution de tels modèles peut devenir prohibitive en réalité ; particulièrement s'il existe beaucoup de scénarios aléatoires à considérer. Il est donc important de tracer un bon compromis entre un modèle simple et soluble et un modèle complexe et fidèle à la réalité.

Alors que les programmes avec recours peuvent servir à corriger ou pénaliser une solution irréalisable, les programmes avec contraintes probabilistes cherchent plutôt à identifier une solution de première étape dont la probabilité de satisfaire certaines contraintes est supérieure à une seuil fixé. Ce deuxième type de modèle met donc l'accent sur la robustesse et la faisabilité de la solution obtenue. Ces programmes peuvent également se révéler utile lorsqu'il est difficile ou impossible d'assigner des pénalités significatives à une solution irréalisable.

Les problèmes avec contraintes probabilistes peuvent se révéler relativement faciles à résoudre dans des cas particuliers. En effet, lorsqu'il existe peu de contraintes probabilistes ou que l'identification d'un petit nombre d'entre elles est suffisant pour obtenir une solution optimale et lorsqu'il est facile de calculer la probabilité d'un évènement particulier, il est possible que ces modèles soient uniquement un peu plus difficiles à résoudre qu'un problème détermi-

niste de grandeur similaire. Par exemple, (Golden et Stewart, 1983) et (Laporte et Louveaux, 1993) ont démontré que sous peu de conditions, les SVRP formulés comme programmes avec contraintes probabilistes pouvaient être réduits à des VRP déterministes équivalents.

On note toutefois que lorsqu'on considère la probabilité de respecter conjointement un ensemble de contraintes (plutôt qu'une seule), le domaine réalisable a de fortes chances de devenir non convexe. Il devient ainsi impossible d'utiliser les techniques d'optimisation linéaire et le problème a de fortes chances de devenir beaucoup plus difficile à résoudre (Kall et Wallace, 1994). En revanche, le domaine réalisable des modèles avec programmation avec recours demeurent convexes si on respecte des conditions relativement faibles sur la fonction de recours et le domaine réalisable (Birge et Louveaux, 1997).

Pour ce mémoire, nous choisissons de considérer un modèle de programmation avec recours, car le VRPSD se prête bien à cette formulation. En particulier, nous considérons qu'il existe un recours suffisamment pertinent qui permet de bien évaluer le coût d'obtenir un échec sur une route. On suppose également qu'il est plus important de mettre l'accent sur cette pénalité que sur la robustesse et la faisabilité des solutions. Par ailleurs on note que les travaux de plusieurs chercheurs ont également porté sur ce type de modèle (Christiansen et Lysgaard, 2007), (Gendreau *et al.*, 1995), (Laporte *et al.*, 2002), (Hjorring et Holt, 1999).

Pour définir la programmation stochastique à deux étapes et recours formellement, on introduit la notation standard tirée de (Birge et Louveaux, 1997). On pose x comme un vecteur de n_1 variables stratégiques de première étape, $\xi(\omega)$ comme le vecteur de variables aléatoires dont la réalisation dépend de l'évènement aléatoire $\omega \in \Omega$ et $y(x, \xi(\omega))$ comme le vecteur de n_2 variables tactiques de seconde étape dépendant de la réalisation des variables aléatoires $\xi(\omega)$ et des variables de première étape x (où n_1 et n_2 représentent des entiers positifs quelconques). On considère également le triplet $(\Omega, \mathscr{T}, P)$ comme un espace de probabilité où Ω représente l'ensemble des réalisations possibles d'une expérience aléatoire, $\mathscr{T}$ représente la collection de tous les sous-ensembles de ces réalisations et P représente la fonction de probabilité. Le processus de décision lorsque l'on considère seulement 2 étapes est bien illustré à la figure 3.1 suivante.

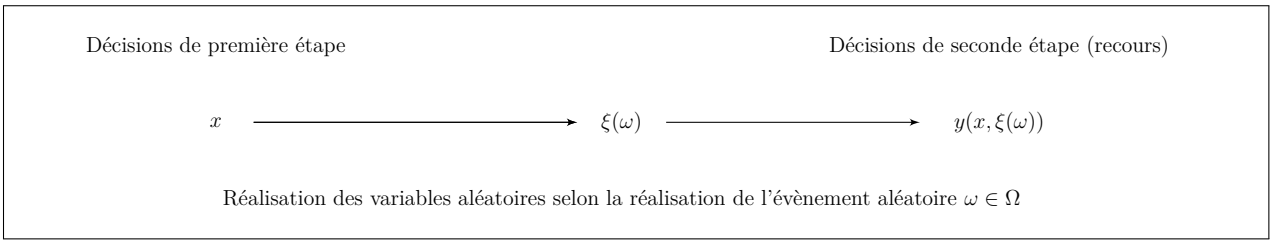

Figure 3.1 Prise de décisions dans un cadre de programmation stochastique à deux étapes avec recours

Par exemple, le problème classique du vendeur de journaux peut être modélisé comme un programme stochastique en deux étapes. Les décisions stratégiques sont représentées par le nombre de journaux à acheter chaque jour alors que les décisions tactiques peuvent être vues comme le nombre de journaux à vendre aux clients ou à retourner à l'imprimeur selon la demande journalière réalisée. Le modèle générique pour la programmation stochastique en nombres entiers et deux étapes peut s'énoncer mathématiquement comme suit (Birge et Louveaux, 1997) :

$$z(x, \xi) = \min_{x \in \mathbb{N}^{n_1}} \{ \, c^T x + \mathbb{E}_\xi \left[Q(x, \xi) \right] : Ax = b \} \tag{3.1}$$

où c, b et A représentent respectivement des vecteurs et matrices fixes de dimension $n_1 \times 1$, $m_1 \times 1$ et $m_1 \times n_1$ (n_1 et m_1 sont des entiers positifs quelconques). $Q(x, \xi)$ représente la valeur de la fonction de recours selon la solution de première étape x et la réalisation des variables aléatoires ξ. Si on définit $\mathcal{Q}(x)$ comme l'espérance de la fonction de recours en fonction de la solution x, alors on obtient :

$$\mathcal{Q}(x) = \mathbb{E}_\xi \left[\min_{y(x,\xi)} q(\xi)^T y(x, \xi) \right] \tag{3.2}$$

$$\text{s. à} \quad Wy(x, \xi) + T(\xi)x = h(\xi) \tag{3.3}$$

$$y(x, \xi) \in \mathbb{N}^{n_2} \tag{3.4}$$

où $y(x, \xi)$ représente le vecteur de n_2 variables de seconde étape. On choisit cette forme pour mettre en évidence la dépendance sur le choix des variables de première étape et la réalisation du vecteur de variables aléatoires ξ. $q(\xi), h(\xi)$ et $T(\xi)$ représentent respectivement des vecteurs et matrices aléatoires de dimension $n_2 \times 1$, $m_2 \times 1$ et $m_2 \times n_1$ (n_2 et m_2 sont des entiers positifs quelconques). W représente quant à elle une matrice fixe de dimension

$m_2 \times n_2$. Comme cette dernière ne dépend pas de la réalisation des variables aléatoires, on dit que le programme est avec recours fixe.

Le modèle initial contient une espérance. Pour contourner cette difficulté, on peut transformer le modèle en son problème déterministe équivalent (« deterministic equivalent problem » - DEP) :

$$z(x, \xi) = \min_{x \in \mathbb{N}^{n_1}} \{ c^T x + \mathcal{Q}(x) : Ax = b\} \tag{3.5}$$

On cherche à minimiser la somme des coûts des variables de première étape et de l'espérance du minimum de la somme du coût des variables de seconde étape. Cette modélisation permet d'obtenir une solution plus robuste que celle obtenue sans considérer les aléas et les distributions des variables aléatoires. Pour bien observer cette propriété, on introduit la notation suivante (Birge et Louveaux, 1997).

On dénote la valeur de la solution du problème (3.5) par RP (« Recourse problem ») pour spécifier qu'il s'agit de la solution optimale lorsque l'on considère la stratégie de recours. On pose également $\bar{x}(\bar{\xi}) = \min_x z(x, \bar{\xi})$ comme la solution optimale du problème lorsque les variables aléatoires prennent leur valeur espérée et on la dénote par EV (« expected value solution »), la solution avec valeur moyenne. On définit alors $EEV = \mathbb{E}_\xi \left[z(\bar{x}(\bar{\xi}), \xi) \right]$ (« Expected result of using expected value ») comme la valeur espérée du problème avec recours (3.5) lorsqu'on utilise la solution $\bar{x}(\bar{\xi})$.

La valeur EEV est de moins bonne qualité que la valeur de la solution obtenue lorsque l'on résout le problème comme un programme stochastique avec recours. Il est possible de prouver formellement que $RP \leq EEV$ dans tous les cas. Intuitivement, la méthode avec recours donne une meilleure solution que EEV, car la seconde méthode est myope et n'utilise pas toute l'information sur la distribution des demandes. À titre d'exemple, si la variance est grande, les valeurs réelles pourraient être très loin des valeurs espérées, ce qui mènerait à une décision largement sous-optimale si on choisit EV. Il est possible d'évaluer l'importance d'ajouter le recours et de considérer l'aléa en calculant $VSS = EEV - RP$ (« Value of the stochastic solution »), la valeur de la solution stochastique.

3.2 Formulation du VRPSD comme un programme stochastique en nombres entiers à deux étapes avec recours

La programmation stochastique en deux étapes avec recours s'applique bien au VRPSD. Dans la première étape, on fixe l'ensemble des routes de manière à couvrir tous les clients exactement une fois et à avoir des routes réalisables. Dans la seconde étape, on visite les clients selon les routes préétablies et on met à jour l'itinéraire à l'aide de la stratégie de *recours* énoncée à la section 1.1. Cette stratégie doit être appliquée de manière à minimiser le coût total d'échec pour la route. Toutefois, comme les demandes sont uniquement révélées lorsque les clients sont visités, la stratégie de *recours* consiste uniquement à *subir* les décisions prises lors de la première étape et à retourner au dépôt lorsque nécessaire.

3.3 Modèle et formulation spécifique

Le modèle proposé pour le VRPSD utilise la notation suivante :

- $\mathcal{N} = \{1, 2, ..., n\}$: l'ensemble des n clients à visiter ;
- $\mathcal{N}' = \mathcal{N} \cup \{o, o'\}$ où o représente le dépôt et o' le puits (copie artificielle du dépôt) ;
- $\mathcal{A} = \{(i,j)|i,j \in \mathcal{N}; i \neq j\} \cup \{(o,j)|j \in \mathcal{N}\} \cup \{(j,o')|j \in \mathcal{N}\}$: l'ensemble des arcs entre les clients, entre les clients et le dépôt, et entre les clients et le puits ;
- $\mathcal{G} = (\mathcal{N}', \mathcal{A})$: le réseau représentant l'ensemble des clients, le dépôt et le puits ainsi que les arcs entre ces derniers ;
- c_{ij} : la distance euclidienne entre $i \in \mathcal{N}'$ et $j \in \mathcal{N}'$. On suppose que le graphe est symétrique. Ainsi : $c_{ij} = c_{ji}\ \forall i,j \in \mathcal{N}'$. De plus, on a $c_{oj} = c_{jo'}\ \forall j \in \mathcal{N}$;
- $\delta^-(S) = \{(i,j) \in \mathcal{A}|j \in S, i \notin S\},\ \forall S \subseteq \mathcal{N}'$: l'ensemble des arcs entrant dans l'ensemble S ;
- $\delta^+(S) = \{(i,j) \in \mathcal{A}|i \in S, j \notin S\},\ \forall S \subseteq \mathcal{N}'$: l'ensemble des arcs sortant de l'ensemble S ;
- $\delta^=(S) = \{(i,j) \in \mathcal{A}|i \in S \subseteq \mathcal{N}', j \in S\},\ \forall S \subseteq \mathcal{N}'$: l'ensemble des arcs ayant leurs deux nœuds dans $S \subseteq \mathcal{N}'$;
- $\mathcal{V}$: l'ensemble des véhicules de capacité identique ;
- Q : la capacité d'un véhicule ;
- ξ_i : la variable aléatoire d'espérance $E[\xi_i]$ indiquant la demande réalisée du client $i \in \mathcal{N}$.

On se rappelle qu'on fait les hypothèses suivantes sur les variables aléatoires ξ :

- $E[\xi_i] \in \mathbb{N}\ \forall i \in \mathcal{N}$: l'espérance des demandes est supposée entière ;
- Les ξ_i sont i.i.d. (indépendantes et identiquement distribuées) ;
- $\xi_i \sim Pois(E[\xi_i])\ \forall i \in \mathcal{N}$: chaque variable suit une distribution de Poisson de paramètre $E[\xi_i]$;
- $1 \leq E[\xi_i] \leq Q\ \forall i \in \mathcal{N}$: l'espérance des demandes est supposée plus petite ou égale à la capacité d'un véhicule.

Tel que mentionné précédemment, toutes ces hypothèses sont acceptables, sauf la première (intégrité de l'espérance des demandes). Dans la section 3.3.2, on explique pourquoi cette hypothèse n'est généralement pas restrictive dans le cas déterministe, mais qu'elle le devient dans le cas stochastique. Celle-ci est néanmoins essentielle à la résolution de notre modèle. Cette prémisse est également couramment utilisée dans la littérature.

Le modèle utilise également les variables de décision suivantes :

◇ Les variables de première étape :
- x_{ij}^v : variable binaire indiquant si le véhicule $v \in \mathcal{V}$ choisit l'arc $(i,j) \in \mathcal{A}$.

◇ Les variables de seconde étape :
- $y_i^v(x,\xi)$: variable entière indiquant le nombre d'échecs se produisant exactement au nœud $i \in \mathcal{N}'$ pour le véhicule $v \in \mathcal{V}$ étant donné x et la réalisation du vecteur ξ.

Bien qu'il soit théoriquement possible d'écrire le problème de second stage $\min_{y(x,\xi)\in\mathbb{N}^{n_2}}\{q(\xi)^T y(x,\xi)\ :\ Wy(x,\xi) + T(\xi)x\ =\ h(\xi)\}$ comme un programme linéaire en nombres entiers, ce modèle serait d'une grande complexité. Plus important encore, ce modèle a peu d'utilité, car une expression analytique de forme fermée est disponible pour $\mathcal{Q}(x)$, la valeur espérée de ce problème, et ce pour tout x, une solution de première étape. Il est donc plus aisé et utile de présenter uniquement la formulation du problème équivalent déterministe du problème maître. La formulation DEP du programme stochastique à deux étapes en nombres entiers du VRPSD peut s'énoncer comme suit :

$$\min_{x} \quad \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}'}\sum_{j\in\mathcal{N}'} c_{ij}x_{ij}^{v} + \mathcal{Q}(x) \tag{3.6}$$

$$(DEP) \qquad \text{s. à :} \quad \sum_{v\in\mathcal{V}}\sum_{(j,i)\in\delta^{-}(\{i\})} x_{ji}^{v} = 1, \quad \forall i \in \mathcal{N} \tag{3.7}$$

$$\sum_{(i,j)\in\delta^{+}(\{i\})} x_{ij}^{v} = \sum_{(j,i)\in\delta^{-}(\{i\})} x_{ji}^{v}, \quad \forall i \in \mathcal{N}, v \in \mathcal{V} \tag{3.8}$$

$$\sum_{(o,i)\in\delta^{+}(\{o\})} x_{oi}^{v} = \sum_{(i,o')\in\delta^{-}(\{o'\})} x_{io'}^{v} = 1, \quad \forall v \in \mathcal{V} \tag{3.9}$$

$$\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} E[\xi_j]x_{ij}^{v} \leq Q \quad \forall v \in \mathcal{V} \tag{3.10}$$

$$x_{ij}^{v} \in \{0,1\} \quad \forall i,j \in \mathcal{N}', \forall v \in \mathcal{V} \tag{3.11}$$

Les contraintes de couverture (4.43) assurent que tous les clients sont visités exactement une fois par chaque véhicule. Les contraintes (4.46) sont nécessaires pour s'assurer qu'on utilise uniquement des routes réalisables en moyenne. Les contraintes (4.44) imposent la conservation du flot aux nœuds pour un véhicule donné. (4.45) spécifient que les véhicules doivent débuter et terminer leur route au dépôt.

$\mathcal{Q}(x)$ représente le coût total espéré d'échec étant donné $x = (x_{ij}^{v})$, une solution de première étape. Cette valeur est donnée par :

$$\mathcal{Q}(x) = \mathbb{E}_{\xi}\left[\min_{y\in Y(x,\xi)} \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}} 2c_{oi}y_{i}^{v}(x,\xi)\right] \tag{3.12}$$

où $Y(x,\xi) = \{y : Wy(x,\xi) + T(\xi)x = h(\xi); y(x,\xi) \in \mathbb{N}^{|\mathcal{V}|}\}$ est le domaine générique réalisable des variables de second stage selon la valeur du vecteur x et la réalisation du vecteur aléatoire ξ selon la notation utilisée dans la section 3.1. Ceci impose que chaque variable $y_{i}^{v}(x,\xi)$ prenne une valeur entière indiquant le nombre d'échecs se produisant exactement au client i pour le véhicule v. Comme les échecs d'un véhicule à l'autre sont indépendants, on peut décomposer la fonction de recours par véhicule et l'écrire comme :

$$\mathcal{Q}(x) = \sum_{v\in\mathcal{V}} \mathcal{Q}^{v}(x^{v}) \tag{3.13}$$

où $\mathcal{Q}^v(x^v) = \mathbb{E}_\xi \left[\min_{y \in Y(x,\xi)} \sum_{i \in \mathcal{N}} 2c_{oi} y_i^v(x,\xi)\right]$ représente le coût total espéré d'échec pour le véhicule v étant donné x^v, le sous-ensemble des arcs traversés par ce véhicule. À cause de l'hypothèse initiale sur la stratégie de recours, on doit s'assurer que pour une route préétablie, un échec se produit à un client donné uniquement si on dépasse la capacité exactement à ce client. On se contente donc uniquement de subir les échecs selon la route décidée préalablement. La minimisation des échecs n'a pas vraiment d'importance, car la valeur des variables de décision de seconde étape est complètement définie par le choix des variables de décision de première étape [1]. On obtient ainsi :

$$\mathcal{Q}^v(x^v) = \sum_{i \in \mathcal{N}} 2c_{oi} \mathbb{E}_\xi \left[y_i^v(x,\xi_i)\right] \tag{3.14}$$

Afin d'obtenir une expression linéaire de la fonction objectif de DEP, on doit calculer la valeur de $\mathbb{E}_\xi \left[y_i^v(x,\xi)\right]$, le nombre espéré d'échecs au client i pour le véhicule v étant donné la solution de première étape x.

3.3.1 Calcul du nombre espéré d'échecs

Bien qu'il puisse sembler difficile de calculer $\mathbb{E}_\xi \left[y_i^v(x,\xi_i)\right]$ a priori, cet exercice est grandement simplifié lorsque l'on considère non seulement l'ensemble des arcs utilisés dans la solution $x = (x_{ij}^v)$, mais également l'ordre de visite des arcs selon les routes empruntées par les véhicules.

Étant donné un véhicule v de capacité Q qui emprunte la route $p = \{i_1 = o, i_2, ..., i_{h-1}, i_h, ..., i_k = o'\} \subseteq \mathcal{N}'$, on définit $m_{i_h}^p = \sum_{l=1}^h \xi_{i_l}$ comme la demande cumulée réalisée au client i_h et $\mu_{i_h}^p = \sum_{l=1}^h E[\xi_{i_l}] = E[m_{i_h}^p]$ comme la demande cumulée espérée le long de ce chemin jusqu'à ce client. Il existe uniquement 2 cas (états) possibles pour le u^e échec au client i_h : dépassement de la capacité du véhicule (échec) ou respect de la capacité (succès). Visuellement, on peut représenter le parcours de la route selon les différentes étapes et états à chaque client (voir figure 3.2).

1. Si la solution de première étape indique exactement dans quel ordre les clients seront visités par un véhicules, alors la minimisation est effectivement inutile. Toutefois, si les routes sont représentées par un ensemble d'arêtes non dirigées, alors il est nécessaire de prendre le minimum des deux orientations possibles pour le route. (voir (Laporte *et al.*, 2002))

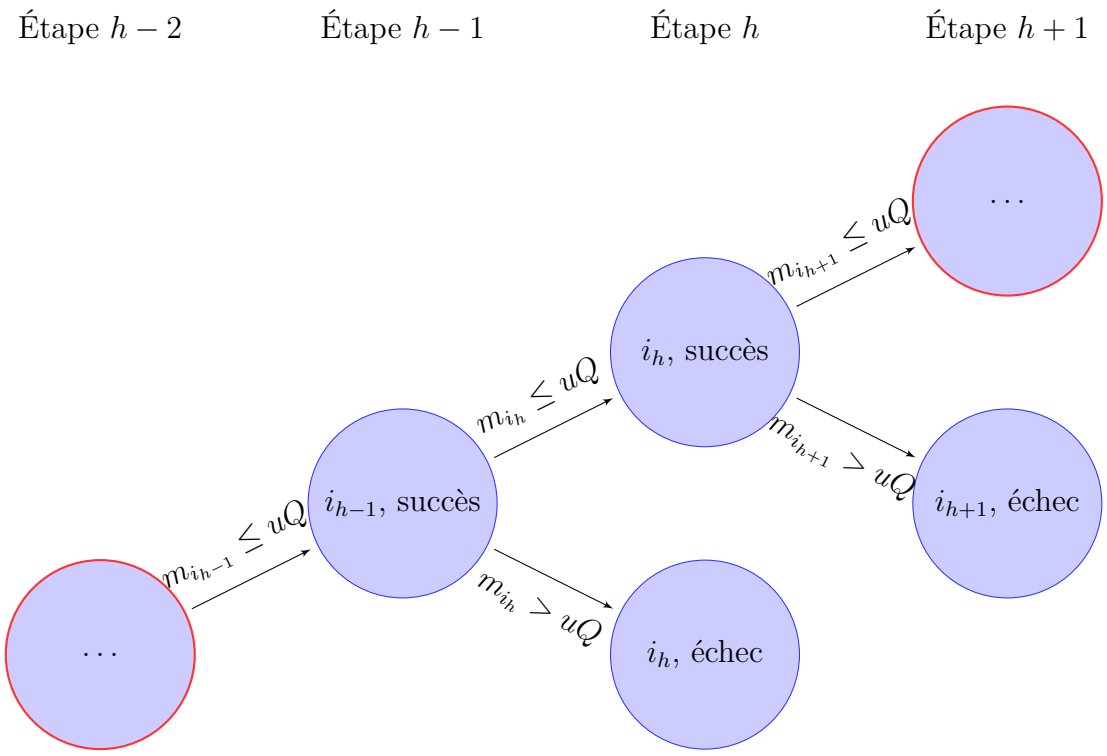

Figure 3.2 Scénarios possibles pour le u^e échec

On sait que la route p visite $i_1, i_2, ..., i_{h-1}$ puis i_h et qu'il est uniquement possible d'obtenir le u^e échec au client i_h si cet échec n'est pas déjà survenu aux clients précédents. En outre, on sait qu'il existe uniquement 2 états mutuellement exclusifs et collectivement exhaustifs à chaque client i. On sait aussi que toutes les demandes sont indépendantes et suivent une distribution de Poisson. Or, on sait que la distribution de Poisson possède la propriété d'additivité. Soit $X(\lambda) \sim Pois(\lambda)$, une variable aléatoire suivant une distribution de Poisson de moyenne λ. Cette propriété implique que si $X_1(\lambda_1) \sim Pois(\lambda_1)$ et $X_2(\lambda_2) \sim Pois(\lambda_2)$ alors $X_3 = X_1 + X_2 \sim Pois(\lambda_1 + \lambda_2)$.

La probabilité d'obtenir le u^e échec au client i_h est donc déterminée par la probabilité que la demande cumulée réalisée au client i_h dépasse la capacité du véhicule sachant qu'elle n'a pas dépassé la capacité au client i_{h-1} précédent. On peut également voir cette probabilité comme la différence entre la probabilité de *ne pas* obtenir le u^e échec *avant* le client i_h et la probabilité de *ne pas* obtenir le u^e échec *exactement* au client i_h. Si on dénote cette probabilité par UFAIL$(\mu^p_{i_h}, i_h, uQ)$, alors on obtient :

$$\text{UFAIL}(\mu^p_{i_h}, i_h, uQ) = P(\Psi(\mu^p_{i_h}) > uQ | \Psi(\mu^p_{i_h} - E[\xi_{i_h}]) \leq uQ) \tag{3.15}$$

$$= P(\Psi(\mu^p_{i_h} - E[\xi_{i_h}]) \leq uQ) - P(\Psi(\mu^p_{i_h}) \leq uQ) \tag{3.16}$$

$$= \sum_{k=0}^{uQ} P(\Psi(\mu^p_{i_h} - E[\xi_{i_h}]) = k) - P(\Psi(\mu^p_{i_h}) = k) \tag{3.17}$$

$$= \sum_{k=0}^{uQ} P(\Psi(\mu^p_{i_{h-1}}) = k) - P(\Psi(\mu^p_{i_h}) = k) \tag{3.18}$$

où $P(\Psi(\lambda) \leq uQ)$ représente la probabilité qu'une variable aléatoire Ψ suivant une distribution de Poisson de paramètre λ prenne une valeur plus petite ou égale à uQ.

Grâce à cette observation, on peut calculer $\text{FAIL}(\mu^p_{i_h}, i)$, le nombre total d'échecs espéré au client i_h sachant que l'on a cumulé une demande espérée de $\mu^p_{i_h}$.

$$\text{FAIL}(\mu^p_{i_h}, i_h) = \sum_{u=1}^{\infty} \text{UFAIL}(\mu^p_{i_h}, i_h, uQ) \tag{3.19}$$

Il peut exister une infinité d'échecs à un client $i \in \mathcal{N}$ donné, car $\xi_i \sim \Psi(E[\xi_i])$ et $\Psi(E[\xi_i])$ est définie sur l'intervalle discret non borné $[0, 1, ..., \infty)$. On utilise donc une convention similaire à celle proposée par de (Christiansen et Lysgaard, 2007) et on remplace ∞ par un nombre suffisamment grand. Plus spécifiquement, on calcule la sommation jusqu'à ce que $\text{UFAIL}(\mu^p_{i_h}, i_h, (u-1)Q) - \text{UFAIL}(\mu^p_{i_h}, i_h, uQ) < 2^{-53}$, car 2^{-53} représente une limite sur la précision machine pour la représentation des nombres flottants.

Le coût espéré d'échec au client i_h sachant qu'on a cumulé une demande espérée de $\mu^p_{i_h}$, dénoté $\text{EFC}(\mu^p_{i_h}, i_h)$ est donc donné par :

$$\text{EFC}(\mu^p_{i_h}, i_h) = 2c_{oi_h}\text{FAIL}(\mu^p_{i_h}, i_h) \tag{3.20}$$

En utilisant l'expression analytique de la fonction de masse d'une variable de Poisson, on peut exprimer le coût total d'échec espéré au client i_h comme :

$$\text{EFC}(\mu^p_{i_h}, i_h) = 2c_{oi_h} \sum_{u=1}^{\infty} \sum_{k=0}^{uQ} \frac{(\mu^p_{i_{h-1}})^k}{e^{(\mu^p_{i_{h-1}})} k!} - \frac{(\mu^p_{i_h})^k}{e^{(\mu^p_{i_h})} k!} \tag{3.21}$$

On suppose que le nombre total d'échecs survenant sur une route donnée est décomposable selon le nombre d'échecs se produisant à chacun des clients visités. Pour chaque route p de o à o' visitant dans l'ordre les arcs $(i_1 = o, i_2), (i_2, i_3), ..., (i_{k-1}, i_k = o')$, on peut ainsi calculer le coût c_p. Par abus de notation, $(i, j) \in p$ indique que l'arc (i, j) est dans la route p si i est visité par la route juste avant le client j. On trouve ainsi :

$$c_p = \sum_{(i,j)\in p} \Big(c_{ij} + \mathrm{EFC}(\mu_j^p, j)\Big) \tag{3.22}$$

où $\mathrm{EFC}(\mu, o) = \mathrm{EFC}(\mu, o') = 0$, car il est impossible d'avoir un échec au dépôt. Ainsi, à partir d'une route p, on est capable de déduire $\mu_j^p,\ \forall (i, j) \in p$ et on peut alors calculer exactement le coût total espéré de cette route.

3.3.2 Création du graphe espace-état $\mathcal{GS}$

Le coût d'une route p dans le graphe $\mathcal{G}$ est donné par :

$$c_p = \sum_{(i,j)\in p} \Big(c_{ij} + \mathrm{EFC}(\mu_j^p, j)\Big) \tag{3.23}$$

La fonction de coût est donc une fonction linéaire de la distance déterministe de chaque arc parcouru et du nombre espéré d'échecs à chaque client. Toutefois, comme expliqué précédemment, afin de déterminer $\mathrm{FAIL}(\mu_j^p, j)$, le nombre espéré d'échecs au client j dans la route p, il est nécessaire de connaître l'espérance de la demande cumulée jusqu'à ce client ainsi que sa demande. Il n'est donc pas possible de calculer le coût d'une route en additionnant le coût de chacun des arcs parcourus dans le graphe $\mathcal{G}$ actuel, car ce réseau ne donne pas d'information sur la demande cumulée.

Afin de contourner cette difficulté, il serait possible de calculer le coût espéré d'échec à chaque client de façon dynamique. Soit l'arc (i, j) dans $\mathcal{G}$. Pour calculer $\mathrm{EFC}(\mu_j^p, j)$, on a uniquement besoin de connaître μ_j^p et c_{oj}. Il serait donc possible de déterminer μ_j^p, la demande cumulée espérée au client j à l'aide de la formulation DEP et du graphe $\mathcal{G}$ actuel. Il suffirait d'ajouter une ressource indiquant cette demande cumulée espérée. Il est important de noter que même si le coût total d'échec à chaque client n'est pas une fonction linéaire de la demande cumulée, on sait que cette fonction croit de façon monotone en μ_j pour tout client j. La section 4.2.4 explique pourquoi le coût total d'un chemin doit être croissant par

rapport à μ_j pour notre implémentation.

Cependant, un tel calcul dynamique nécessite le calcul de 3.21. Il pourrait donc se révéler coûteux en temps de calcul, même si on s'assure de calculer une probabilité d'échec selon une demande cumulée seulement une fois. Nous choisissons donc d'adopter la stratégie de (Christiansen et Lysgaard, 2007) et d'ajouter l'information sur la demande cumulée explicitement dans un graphe espace-état auxiliaire. Ce graphe $\mathcal{GS} = (\mathcal{NS}', \mathcal{AS})$ inclut la valeur de μ_i au nœud i. Le « $\mathcal{S}$ » dans $\mathcal{GS}$ indique qu'il s'agit d'un graphe d'état (« state graph » en anglais). En effet, on peut voir chaque nœud $\mathscr{V}(\mu, i) \in \mathcal{NS}$ comme un état particulier au client i et le coût de chaque arc comme un coût de transition d'un état à un autre. On sera ainsi en mesure de fixer un coût espéré d'échec fixe pour chacun des arcs. Le graphe $\mathcal{GS}$ contient les nœuds et les arcs suivants :

- Un nœud origine os et un nœud puits os' ;
- Pour chaque client $i \in \mathcal{N}$ et pour tout $\mu \in \{1, 2, ..., Q\}$, un nœud $\mathscr{V}(\mu, i)$;
- Pour chaque paire de clients $i, j \in \mathcal{N}$ et pour tout $\mu \in \{1, 2, ..., Q - E[\xi_j]\}$, un arc allant de $\mathscr{V}(\mu, i)$ à $\mathscr{V}(\mu + E[\xi_j], j)$ de coût $c_{ij} + \text{EFC}(\mu + E[\xi_j], j)$;
- Pour chaque client $i \in \mathcal{N}$, un arc de os à $\mathscr{V}(E[\xi_i], i)$ de coût $c_{oi} + \text{EFC}(E[\xi_i], i)$;
- Pour chaque client $i \in \mathcal{N}$ et pour tout $\mu \in \{1, 2, ..., Q\}$, un arc de $\mathscr{V}(\mu, i)$ à os' de coût $c_{io'}$.

On obtient le graphe illustré à la figure 3.3 :

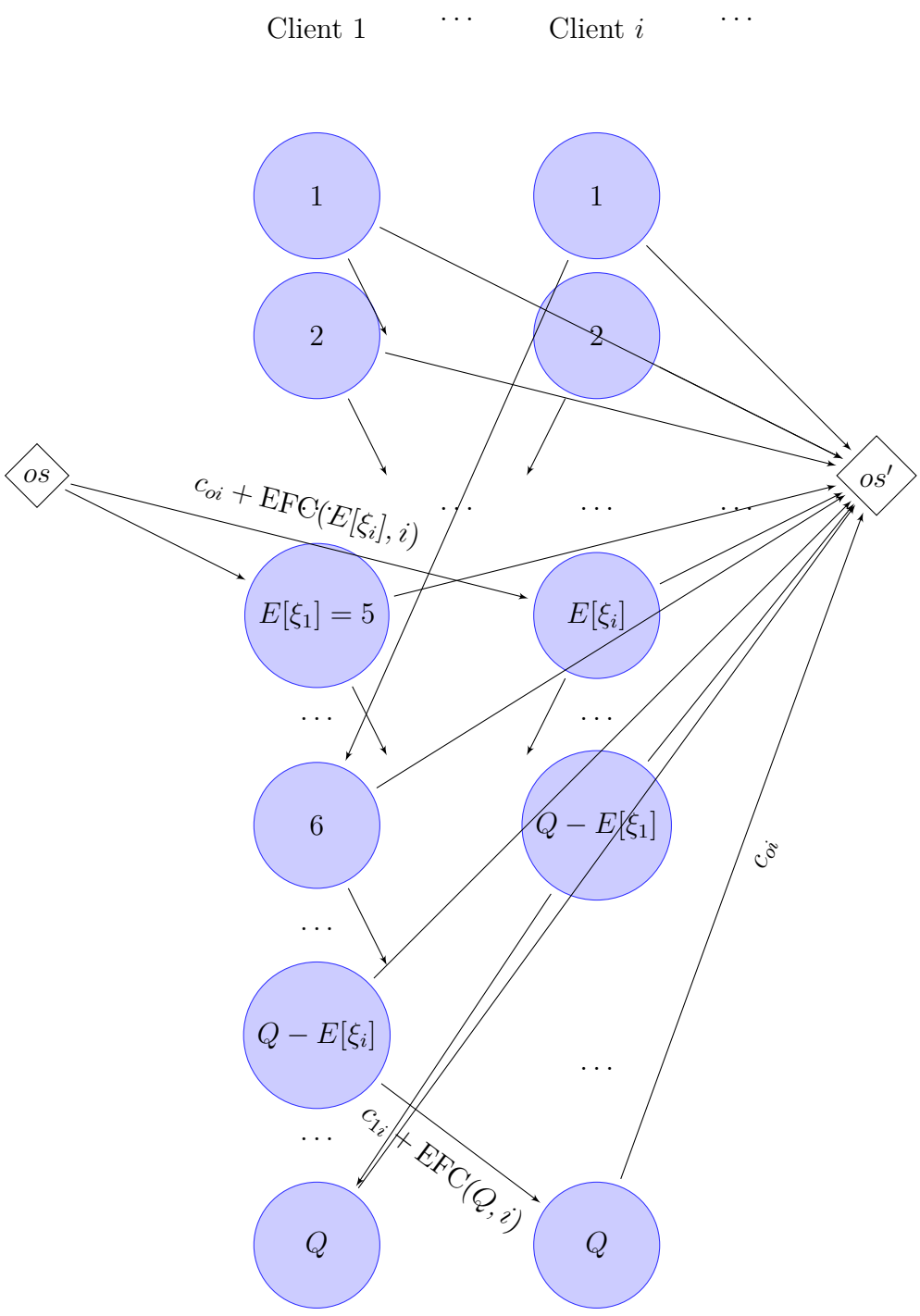

Figure 3.3 Extrait du graphe espace-état $\mathcal{GS}$

On se rappelle qu'on suppose que chacune des demandes espérées $E[\xi_i]$ peut uniquement prendre des valeurs entières et que l'on considère uniquement les routes réalisables en moyenne, c'est-à-dire les routes p avec $\mu_j^p \leq Q,\ \forall (i,j) \in p;\ i,j \in \mathcal{N}$. Comme on crée un nœud $\mathscr{V}(\mu, i)$ pour tout $\mu \in \{1, 2, ..., Q\}$ et $i \in \mathcal{N}$, on discrétise les demandes cumulées espérées réalisables à chaque client de 1 à Q.

On note que dans le cas d'un VRP déterministe, il n'est pas nécessaire de supposer l'intégrité de l'espérance des demandes, car il serait possible d'augmenter la précision en utilisant une discrétisation plus fine. Si $E[\xi_i] \notin \mathbb{N}$, on pourrait notamment choisir la demande espérée dans l'ensemble $\{E[\xi_i^t] = 1 + t\eta : \eta \in \mathbb{Q}^+, t \in \mathbb{N}, E[\xi_i^t] \leq Q\}$; et ce pour tout $i \in \mathcal{N}$ où t peut être vue comme le pas et η comme la plus petite précision désirée (par exemple $\eta = 10^{-2}$ si on a des demandes réelles arrondies à 2 décimales). Il serait donc encore possible de représenter les demandes cumulées espérées à chaque noeud à l'aide d'une discrétisation comme on le fait dans $\mathcal{GS}$.

Toutefois, lorsqu'on considère le cas stochastique avec une distribution de Poisson, on ne peut faire cette hypothèse, car cette distribution est définie sur les entiers positifs. Il serait notamment possible d'utiliser une distribution normale si on souhaite considérer des demandes réelles. Cependant, comme cette distribution utilise deux paramètres plutôt qu'un seul pour la distribution de Poisson, la taille du graphe espace-état serait susceptible d'exploser. Par ailleurs, comme on souhaite comparer nos résultats avec ceux de (Christiansen et Lysgaard, 2007), on suppose que les demandes sont entières et suivent une distribution de Poisson.

Comme $\mathcal{GS}$ contient $O(nQ)$ noeuds et $O(n^2Q)$ arêtes, sa taille peut exploser pour de plus grandes instances. Il est possible de réduire sa taille en éliminant les arêtes et les nœuds non atteignables. Ceci peut être réalisé en calculant $\tilde{\mu}(i) \subseteq \{1, 2, ..., Q\}$, l'ensemble des demandes cumulées espérées réalisables pour le client i.

Pour ce faire, on peut résoudre un problème de somme des sous-ensembles (« subset-sum problem ») qui indique s'il est possible d'atteindre une certaine somme avec un sous-ensemble d'éléments. Plus spécifiquement, on tente de déterminer quels entiers dans le vecteur $I(i) = (1, 2, ..., Q - E[\xi_i])$ peuvent être obtenus par la somme partielle de chacun des sous-ensembles du vecteur $D(i) = (E[\xi_1], E[\xi_2], ..., E[\xi_{i-1}], E[\xi_{i+1}], ..., E[\xi_n])$, le vecteur de demandes espérées ne contenant pas $E[\xi_i]$. On note que les demandes espérées cumulées sont répétées si elles correspondent à des clients différents.

Bien que ce problème soit $\mathcal{NP}$-difficile (Cormen *et al.*, 2001) (Garey et Johnson, 1979), il existe un algorithme pseudo-polynomial permettant de le résoudre en temps raisonnable, car la capacité Q est bornée et relativement petite. On résout ce problème en créant dynamiquement n matrices $M^{(n-1)\times(Q-E[\xi_i]+1)}$ de valeurs booléennes pour chaque client i. À la fin de l'algorithme, on saura qu'il est possible d'atteindre une demande cumulée de μ au client i si et seulement si l'élément $M_{(n-1),\mu-E[\xi_i]}$ contient 1 (où $E[\xi_i] \leq \mu \leq Q - E[\xi_i]$). On crée chacune des n matrices en $O(nQ)$.

Pour le client i, la matrice M est obtenue en calculant dynamiquement les valeurs du tableau avec la récursion suivante :

$$M_{j,0} = 1 \qquad \forall j \in \{1, ..., n-1\} \tag{3.24}$$

$$M_{j,s} = \begin{cases} 1 \text{ si } M_{j-1,s} = 1 \text{ ou } M_{j-1,s-D(i)_j} = 1 \text{ et } s \geq D(i)_j \\ 0 \text{ sinon} \end{cases} \qquad \forall j \in \{1, ..., n-1\}\ , \forall s \in I(i) \tag{3.25}$$

Comme on le voit à la section 4.2.10, l'algorithme de génération de colonnes ne sera pas considérablement ralenti par l'existence de nombreux nœuds inutiles, car l'algorithme de résolution du sous-problème ne les considèrera pas. Toutefois, la résolution du problème de somme de sous-ensembles peut se révéler crucial lorsque le nombre de nœuds devient si grand qu'il est impossible d'allouer assez de mémoire pour lire et sauvegarder le graphe. Ceci est notamment le cas pour l'instance E-n33-k4 dont les véhicules ont une capacité de 8000 et qui contient 32 clients. Même si on considère uniquement les nœuds $\mathscr{V}(\mu, i)$ avec $\mu \geq E[\xi_i]$, sans élimination des nœuds inutiles, le graphe $\mathcal{GS}$ contiendrait 226 663 nœuds et 6 133 247 arrêtes. À titre de comparaison, l'instance P-n55-k7 (l'instance avec le plus grand nombre de nœuds dans le graphe espace-état si on ne considère pas E-n22-k4 et P-n22-k8, les autres instances avec une très grande capacité) contient 8 193 nœuds et 387 196 arrêtes. Avec l'algorithme de réduction, E-n33-k4 contient cependant 17 164 nœuds et 462 075 arêtes (une facteur de réduction de plus de 13).

Tel que décrit dans la section des résultats, les instances avec une très grande capacité et plusieurs clients ne seraient pas solubles sans l'application de cet algorithme. De plus, lorsqu'on considère un VRP déterministe, cet algorithme permet réellement de discrétiser les demandes fractionnaires en utilisant une précision très fine tout en s'assurant que le modèle demeure soluble. L'algorithme d'élimination des nœuds inutiles devient donc très utiles pour augmenter la flexibilité du modèle et relâcher certaines hypothèses sur les demandes

des clients.

Suite à la création de $\mathcal{GS}$ et l'élimination des nœuds inutiles, on obtient le graphe espace-état $\mathcal{GS} = (\mathcal{NS}', \mathcal{AS})$ où $\mathcal{NS}' = \mathcal{NS} \cup \{os, os'\}$ et on ne considère que les demandes espérées cumulées possibles à chaque client. Comme toute route réalisable dans le graphe $\mathcal{G}$ original correspond à exactement une route dans le graphe auxiliaire $\mathcal{GS}$ de coût identique, on peut reformuler DEP en fonction de $\mathcal{GS}$. On obtient ainsi le programme $DEPs$ suivant :

$$\min_{\hat{x}} \quad \sum_{v \in \mathcal{V}} \sum_{a \in \mathcal{NS}'} \sum_{b \in \mathcal{NS}'} \hat{c}_{ab} \hat{x}^v_{ab} \tag{3.26}$$

$$(DEPs) \qquad \text{s. à} \quad \sum_{v \in \mathcal{V}} \sum_{(b,a) \in \delta^-(C(i))} \hat{x}^v_{ba} = 1, \quad \forall i \in \mathcal{N} \tag{3.27}$$

$$\sum_{(a,b) \in \delta^+(\{a\})} \hat{x}^v_{ab} = \sum_{(b,a) \in \delta^-(\{a\})} \hat{x}^v_{ba}, \quad \forall a \in \mathcal{NS}, \forall v \in \mathcal{V} \tag{3.28}$$

$$\sum_{(os,a) \in \delta^+(\{os\})} \hat{x}^v_{os,a} = \sum_{(a,os') \in \delta^-(\{os'\})} \hat{x}^v_{a,os'} = 1, \quad \forall v \in \mathcal{V} \tag{3.29}$$

$$\hat{x}^v_{ab} \in \{0,1\}, \quad \forall a, b \in \mathcal{NS}' \tag{3.30}$$

où $C(i) = \{\mathscr{V}(\mu, i) \in \mathcal{NS} \mid \mu \in \tilde{\mu}(i)\}$ représente l'ensemble des nœuds de $\mathcal{NS}$ associés au client i atteignables avec une demande cumulée espérée réalisable et $\delta^-(I), \delta^+(I)$ représentent respectivement l'ensemble des arcs entrant et sortant de l'ensemble I dans le graphe $\mathcal{GS}$. On pourrait écrire $\delta^-_{\mathcal{GS}}(I), \delta^+_{\mathcal{GS}}(I)$ et $\delta^-_{\mathcal{G}}(I), \delta^+_{\mathcal{G}}(I)$ pour différencier les ensembles dans $\mathcal{G}$ et $\mathcal{GS}$, mais on omet l'indice par souci de parcimonie lorsque le contexte est clair. Les variables $\hat{x}^v_{ab}$ représentent le flot passant sur l'arc $(a, b) \in \mathcal{AS}$. On note que $\sum_{(a,b) \in \mathcal{AS} \,:a \in C(i); b \in C(j)} \hat{x}^v_{ab} = x^v_{ij}$. Dans la suite de ce mémoire, on utilisera les indices a, b ou u, v pour indiquer des nœuds dans $\mathcal{NS}$ et les indices i, j pour indiquer les nœuds dans $\mathcal{N}$.

$\hat{c}_{ab}$ représente le coût modifié des arcs de $\mathcal{AS}$ tel que décrit dans la construction du graphe $\mathcal{GS}$. $\hat{c}_{ab}$ est donc le coût total espéré de déplacement du client associé à a (ou de la source) jusqu'au client associé à b (ou le puits). Les contraintes (3.27) , (3.28) et (3.29) jouent essentiellement le même rôle que les contraintes (4.43), (4.44) et (4.45) correspondantes dans DEP. On note que la contrainte de capacité (4.46) est implicite dans la structure du graphe $\mathcal{GS}$ correspondant.

On note que pour le véhicule v empruntant la route p dans $\mathcal{G}$, le coût de la route est donné par :

$$c_p = \sum_{\substack{(i,j)\in p \\ a\in C(i) \\ b\in C(j)}} \hat{c}_{ab} \tag{3.31}$$

$$= \sum_{(i,j)\in p} \Big(c_{ij} + \mathrm{EFC}(\mu_j^p, j)\Big) \tag{3.32}$$

$$= \sum_{(i,j)\in p} \Big(c_{ij} + 2c_{oj}\mathrm{FAIL}(\mu_j^p, j)\Big) \tag{3.33}$$

Ceci est cohérent avec l'équation (3.23). Le coût des arcs dans $\mathcal{GS}$ représente donc bien le coût total espéré de se déplacer d'un client à un autre avec une demande cumulée espérée donnée.

Si on fait abstraction du coût espéré d'échec, on observe que $DEPs$ représente un VRP déterministe. Comme on le démontre dans la section suivante, ce type de problème se prête particulièrement bien à la décomposition de Dantzig-Wolfe et la génération de colonnes.

3.4 Décomposition de Dantzig-Wolfe

Tel qu'illustré à la section précédente, on peut décomposer le coût total d'échec ainsi que le domaine réalisable des variables de seconde étape par véhicule. Même si on ne présente pas la formulation explicite du programme stochastique en nombres entiers à deux étapes explicitement, on sait qu'il possède la structure suivante (selon la notation utilisée à la section 3) :

$$\begin{array}{|c|ccc|c|c|c|}
\hline
A & & & & x & = & b \\
\hline
D_{v_1} & & & & & = & e_{v_1} \\
\cdots & & & & & & \vdots \\
 & D_{v_V} & & & & = & e_{v_V} \\
T_{v_1} & \cdots & W & & y & = & h_{v_1} \\
\cdots & & \cdots & \cdots & & = & \vdots \\
 & T_{v_V} & & W & & = & h_{v_V} \\
\hline
\end{array} \tag{3.34}$$

En utilisant l'expression fermée de $\mathcal{Q}(x)$, on peut calculer le coût espéré d'échec pour toute solution de première étape. On considère donc implicitement les variables de seconde étape dans la fonction objectif et on omet ces dernières ainsi que les contraintes de la forme $Tx + Wy = h$ dans dans le problème équivalent déterministe. Comme on utilise le graphe $\mathcal{GS}$ pour résoudre $DEPs$, ce problème présente alors la forme :

$$\begin{array}{|c|} \hline \hat{A} \\ \hline \hat{D}_{v_1} \\ \hline \end{array} \quad \begin{array}{|c|} \hline \hat{x} \\ \hline \\ \hline \end{array} \quad \begin{array}{c} = \\ = \\ \\ = \end{array} \quad \begin{array}{|c|} \hline \hat{b} \\ \hline \hat{e}_{v_1} \\ \hline \end{array} \qquad \begin{array}{|c|} \hline \hat{D}_{v_V} \\ \hline \end{array} \quad \begin{array}{|c|} \hline \hat{e}_{v_V} \\ \hline \end{array} \tag{3.35}$$

On peut donc écrire $DEPs$ comme $\{\min_{\hat{x}} \hat{c}^T \hat{x} | \hat{A}\hat{x} = \hat{b},\ \hat{D}\hat{x} = \hat{e},\ \hat{x} \in \{0,1\}^{|\mathcal{AS}||\mathcal{V}|}\}$ où $\hat{c}$ représente le vecteur de coût tenant compte du coût total espéré d'échec. $\hat{A}\hat{x} = \hat{b}$ représente les contraintes liantes ou difficiles associées aux contraintes (3.27) dans $DEPs$ et $\hat{D}\hat{x} = \hat{e}$ représente les contraintes faciles associées aux contraintes (3.28) à (3.29) de $DEPs$.

La matrice $\hat{A} \in \mathbb{N}^{|\mathcal{N}|\times|\mathcal{AS}||\mathcal{V}|}$ possède le même nombre de lignes que $A \in \mathbb{N}^{|\mathcal{N}|\times|\mathcal{A}||\mathcal{V}|}$, mais beaucoup plus de colonnes (un facteur de $O(Q)$ par rapport à A). Pour chaque véhicule, la matrice $\hat{D}^v \in \mathbb{N}^{|\mathcal{NS}'|\times|\mathcal{AS}|}$ contient également un facteur de $O(Q)$ lignes et colonnes additionnelles par rapport à $D^v \in \mathbb{N}^{|\mathcal{N}'|\times|\mathcal{A}|}$ correspondants dans DEP. Bien que la taille de ces matrices augmente, la formulation $DEPs$ permet d'incorporer directement le coût espéré d'échec dans le graphe sous-jacent et on peut éliminer toutes les contraintes de la forme $T^v x^v + Wy^v = h^v$.

On observe également que $DEPs$ présente une structure bloc angulaire où l'on peut séparer les contraintes liantes de la forme $A\hat{x} = \hat{b}$ des contraintes faciles. De plus, les contraintes faciles peuvent être regroupées par véhicule v. Selon le principe de décomposition de Dantzig-Wolfe (Desrosiers et Lubbecke, 2005), on peut donc décomposer $DEPs$ en un problème maître PM et un sous-problème SP. Sous sa forme générique, le problème maître peut s'écrire comme :

$$\min_{x} \quad \sum_{v\in\mathcal{V}}\sum_{p\in\Lambda_v} \hat{c}_p\lambda_p \tag{3.36}$$

$$(PM') \qquad \text{s. à :} \quad \sum_{v\in\mathcal{V}}\sum_{p\in\Lambda_v} \alpha_{ip}\lambda_p = 1 \ \forall i \in \mathcal{N}. \tag{3.37}$$

$$\lambda_p \in \{0,1\} \ \forall v \in \mathcal{V}, p \in \Lambda_v \tag{3.38}$$

où Λ_v représente l'ensemble des points extrêmes du domaine $\Sigma_v = \{\hat{x}^v : \hat{D}^v\hat{x}^v = \hat{e}^v\}$ défini par le sous-ensemble des contraintes de $DEPs$ (3.28) à (3.29) associées au véhicule v. Le sous-problème possède la propriété d'intégrité, ce qui veut dire que la solution optimale du sous-problème est entière. Ainsi, les points extrêmes des domaines $\{\hat{x}^v : \hat{D}^v\hat{x}^v = \hat{e}^v\}$, et $\{\hat{x}^v : \hat{D}^v\hat{x}^v = \hat{e}^v, \hat{x}^v \in \mathbb{N}^{|\mathcal{AS}|}\}$ sont identiques. Chaque point extrême du domaine Σ_v correspond à un chemin entier de la source os au puits os' emprunté pas le véhicule v qui visite un ensemble d'arcs $(i,j) \in \mathcal{A}$ et respecte les contraintes de conservation de flot et de capacité. $\hat{c}_p = \sum_{(i,j)\in p} \hat{c}_{ij}$ représente le coût d'un tel chemin.

La variable $\lambda_p = 1$ si on choisit ce chemin et $\lambda_p = 0$ sinon. (3.38) impose donc que l'on choisisse une combinaison convexe des points extrêmes du domaine Σ_v si on choisit le véhicule v. La combinaison de (3.38) et de la propriété d'intégrité assure que les variables x^v_{ij} originales seront entières et respectent la contrainte (3.30). En effet, on peut toujours retrouver les variables x^v_{ij} originales à l'aide de la transformation $x^v_{ij} = \sum_{p\in\Lambda_v} \lambda_p\alpha_{ijp}$ où α_{ijp} indique le nombre de fois qu'un véhicule qui choisit la route p emprunte l'arc (i,j). On note que $\sum_{(b,a)\in\delta^-_{\mathcal{GS}}(C(j));b\in C(i)} \alpha_{bap} = \alpha_{ijp}$ et $\sum_{j\in\delta^-_{\mathcal{G}}(\{j\})} \alpha_{ijp} = \alpha_{jp}$.

De plus, le paramètre $\alpha_{ip} = 1$ indique le nombre de fois que le chemin p visite le client $i \in \mathcal{N}$. Les contraintes (3.37) assurent donc que chaque client sera visité exactement une fois. Comme les véhicules sont identiques, il suffit de considérer un seul ensemble de points extrêmes Λ correspondant à un des ensembles Λ_v. On obtient ainsi le problème maître suivant :

$$\min_{x} \quad \sum_{p\in\Lambda} \hat{c}_p\lambda_p \tag{3.39}$$

$$(PM) \qquad \text{s. à :} \quad \sum_{p\in\Lambda} \alpha_{ip}\lambda_p = 1, \quad \forall i \in \mathcal{N} \tag{3.40}$$

$$\lambda_p \in \{0,1\}, \ \forall p \in \Lambda \tag{3.41}$$

Comme il existe un très grand nombre de routes réalisables, mais que seulement quelques-unes de ces dernières sont utilisées dans la solution optimale du problème maître, il est inutile et inefficace de toutes les énumérer. À l'itération k, l'algorithme de génération de colonnes fait donc appel à PMR_k, le problème maître restreint qui considère $\Lambda_k \subset \Lambda$, un sous-ensemble restreint des routes réalisables de os à os'.

Afin d'identifier de nouvelles routes $p \notin \Lambda_k$ dans $\mathcal{GS}$ permettant d'améliorer la solution actuelle du problème maître, on résout un sous-problème SP_k qui identifie les routes réalisables de coût réduit strictement négatif. Pour ce faire, il faut modifier le coût des arcs du graphe $\mathcal{GS}$ afin de refléter leur coût réduit par rapport à la solution optimale actuelle du problème maître réduit. Étant donné $\lambda \in \mathbb{R}^{|\Lambda_k|}$, une solution primale réalisable de la relaxation linéaire de PMR_k, on est capable de calculer une solution duale optimale $\pi \in \mathbb{R}^m$ où les élément du vecteur π sont associés à une contrainte (3.40) et donc à un client. On peut ainsi résoudre SP_k, qui s'énonce comme :

$$(SP_k) \qquad \min_{p \in \Lambda \setminus \Lambda_k} \hat{c}_p - \pi_k^T \mathscr{A}_p \tag{3.42}$$

où $\bar{c}_p = \hat{c}_p - \pi_k^T \mathscr{A}_p = \sum_{(i,j)\in p; a\in C(i), b\in C(j)} (\hat{c}_{ab} - \pi_{jk}) \hat{x}_{ab}^p$ représente le coût réduit d'une route $p \in \Lambda$ et $(i,j) \in p$ indique que l'arc $(i,j) \in \mathcal{A}$ appartient à p. $\mathscr{A}_p$ représente la colonne associée à ce chemin. On ajoute cette colonne ainsi que la variable de décision λ_p correspondante à PMR_k si $\bar{c}_p < 0$.

SP_k correspond à un problème de plus court chemin avec contraintes de ressources (« shortest path problem with ressource constraints » - SPPRC) où le coût sur les arcs correspond à leur coût réduit par rapport à la solution optimale de PMR_k. Bien que les contraintes de ressources ne soient pas explicites dans la formulation de $DEPs$, on note que l'utilisation du graphe sous-jacent $\mathcal{GS}$ implique que les routes sont réalisables en moyenne, c.-à-d. que la demande cumulée ne dépasse pas la capacité du véhicule. On remarque également que trouver un plus court chemin dans $\mathcal{GS}$ est équivalent à trouver un plus court chemin dans $\mathcal{G}$ dont la demande cumulée espérée respecte les contraintes de capacité. La complexité théorique est la même pour ces deux problèmes. On peut formuler le SPPRC à l'itération k comme suit :

$$\min_{\hat{x}} \quad \sum_{\substack{a\in\mathcal{NS}' \\ a\in C(i)}} \sum_{\substack{b\in\mathcal{NS}' \\ b\in C(j)}} (\hat{c}_{ab} - \pi_{jk})\hat{x}_{ab} \tag{3.43}$$

$$(SP_k(non-elem)) \qquad \text{s. à.} \quad \sum_{(a,b)\in\delta^+(\{a\})} \hat{x}_{ab} = \sum_{(b,a)\in\delta^-(\{a\})} \hat{x}_{ba} \quad \forall a \in \mathcal{NS} \tag{3.44}$$

$$\sum_{(os,a)\in\delta^+(\{os\})} \hat{x}_{os,a} = \sum_{(a,os')\in\delta^-(\{os'\})} \hat{x}_{a,os'} = 1 \tag{3.45}$$

$$\hat{x}_{ab} \in \{0,1\} \quad \forall a,b \in \mathcal{NS}' \tag{3.46}$$

Les contraintes (3.44) assurent la conservation de flot sur les arcs incidents à un noeud donné alors que (3.45) impose le départ et l'arrivée à partir d'un dépôt. Pour un problème de plus court chemin (avec ou sans contraintes de ressources) avec des coûts strictement positifs sur les arcs, les contraintes (3.44) et (3.45) sont suffisantes pour s'assurer que la solution ne comprend pas de sous-tours (cycles n'incluant pas la source ou le puits) ou de routes avec des cycles. La solution optimale correspond donc bien à un chemin élémentaire de os à os'. En effet, avec des coûts strictement positifs sur les arcs, si une solution possède des sous-tours ou des cycles, elle ne peut être optimale, car on obtient une solution réalisable de coût strictement plus petit en supprimant ce sous-tour. Cependant, ce n'est pas nécessairement le cas pour notre sous-problème, car on considère le coût réduit des arcs.

On appelle le sous-problème $SP_k(non-elem)$ pour mettre l'accent sur le fait qu'on peut générer des routes visitant un client plus d'une fois. En effet, bien que la contrainte (3.46) indique qu'un arc peut uniquement être emprunté une fois, il est possible que la route revisite un client plus d'une fois avec différentes demandes cumulées. Ceci est équivalent à poser $\Lambda = \mathscr{P}$, l'ensemble des routes de os à os' réalisables (pouvant passer plusieurs fois par un client donné avec différentes demandes cumulées espérées).

Même si on considère $\mathscr{P}_k$ et non $\mathscr{PE}_k$, le sous-ensemble des routes de $\mathscr{P}$ passant exactement une fois par chaque client, chaque route dans la solution optimale de PM visitera exactement une fois chaque client. En effet, la contrainte (3.40) du problème maître assure que seulement des routes élémentaires pourront être utilisées dans la solution. Même si l'on permettait l'utilisation de routes non élémentaires dans la solution de PM, une solution entière qui couvre un client plus d'une fois ne serait pas optimale pour ce programme. Avec des routes entières, il est effectivement possible de couvrir un client plus d'une fois dans deux situations : deux routes différentes couvrent le client ou une route possède un cycle.

Ces deux cas sont sous-optimaux pour PM, car $c_{ij} > 0 \; \forall (i,j) \in \mathcal{A}$ et les coûts de déplacement respectent l'inégalité du triangle. Par ailleurs $\text{EFC}(\mu_j, j)$ est croissante par rapport à μ_j (voir la preuve en annexe). Il n'est donc pas optimal qu'une solution sur-couvre certains clients (visite un client plus d'une fois).

On en conclut qu'on pourrait changer l'égalité de la contrainte (3.40) par $\geq$ et conserver la solution optimale initiale de PM. Les travaux de (Desrochers *et al.*, 1992) expliquent pourquoi cette technique peut se révéler bénéfique. Elle permet effectivement de stabiliser la valeur des variables duales et ainsi d'augmenter la stabilité numérique de l'algorithme de génération de colonnes. De plus, elle permet d'inclure des variables de décision associées à des routes non élémentaires à des bases du problème maître.

Les sections suivantes présentent l'algorithme général de BCP ainsi que l'algorithme utilisé pour résoudre le sous-problème.

CHAPITRE 4

MÉTHODE DE RÉSOLUTION

4.1 Résolution par génération de colonnes et ajout de plans coupants dans un contexte d'énumération implicite - « branch-cut-and-price » - BCP

Il est possible de résoudre $DEPs$ directement à l'aide d'une formulation explicite à trois indices sur les variables de flot sur les arcs $\hat{x}^v_{ab}$. Cette formulation est dite compacte ou non décomposée par opposition à la formulation de Dantzig-Wolfe qui se base sur un problème maître et un sous-problème. Ce modèle peut être résolu à l'aide d'un solveur commercial pour les programmes mathématiques linéaires en nombres entiers comme `CPLEX` de la compagnie IBM.

Cependant, la symétrie du problème et la faiblesse de la relaxation linéaire de $DEPs$ sont susceptibles de rendre la résolution de certaines instances impossible avec la formulation compacte en un temps « raisonnable ». Nous choisissons donc d'exploiter la structure bloc-angulaire de $DEPs$ et de décomposer le problème initial en son problème maître PM et son sous-problème SP. Nous considérons également l'ajout d'inégalités valides pour renforcer la relaxation linéaire du problème maître et nous imbriquons le tout dans un arbre de branchement pour générer des solutions entières. Ceci donne lieu à l'algorithme itératif de génération de colonnes avec plans coupants (BCP) décrit dans les paragraphes suivants.

On cherche à obtenir la solution optimale de PM, c.-à.-d. une solution de PM entière et réalisable de valeur minimale. Comme PM considère $\mathscr{P}$, l'ensemble des routes réalisables, il contient un nombre exponentiel de variables de décision. Il est donc impossible de le résoudre directement. Par ailleurs, il est inutile d'énumérer $\mathscr{P}$, car seulement quelques-unes de ces routes sont utilisées dans la solution optimale de PM. On utilise donc plutôt PMR^l_k, le problème maître restreint à l'itération k du l^e nœud de branchement qui considère seulement $\mathscr{P}^l_k \subseteq \mathscr{P}$, un sous-ensemble des routes réalisables.

Afin de déterminer $\mathscr{P}^l_k$, on procède comme suit. Pour tout nœud de branchement l et à chaque itération k de l'algorithme de génération de colonnes, on résout la relaxation linéaire de PMR^l_k à l'aide d'un solveur linéaire commercial comme `CPLEX`. On utilise ensuite les valeurs des variables duales fournies par la solution optimale de ce problème pour modifier le

poids des arcs dans le graphe $\mathcal{GS}$ pour refléter leur coût réduit. On peut ainsi résoudre SP_k, le sous-problème de plus court chemin avec contraintes de ressources sur le graphe $\mathcal{GS}$ dont le coût des arcs est modifié. La résolution de ce sous-problème permet d'identifier les routes de coût réduit strictement négatif à l'itération k, c.-à.-d. celles qui permettent d'améliorer la solution actuelle de PMR_k^l.

Après avoir résolu le sous-problème, on ajoute un certain nombre de variables de décision binaires à PMR_k^l correspondant aux chemins de coût réduit négatif identifiés par l'algorithme de plus court chemin. On ajoute également une colonne à PMR_k^l correspondant à chacune de ces routes et on incrémente le compteur d'itération. On répète cette procédure itérative à chaque nœud de branchement jusqu'à ce que la valeur du plus court chemin fournie par la résolution du sous-problème soit plus grande ou égale à zéro.

Il est possible que la solution optimale de PMR_k^l fournie à la fin de l'algorithme de génération de colonnes à un nœud de branchement soit fractionnaire. Comme le problème maître original est en nombres entiers, on doit toutefois s'assurer d'obtenir une solution entière. Pour ce faire, il est possible d'identifier des inégalités respectées pour toute solution entière, mais qui sont violées par la solution fractionnaire actuelle. On crée un nouveau nœud de branchement L_{a+1} ayant L_l comme parent et on ajoute ces inégalités valides ou plans coupants à PMR_0^{a+1} avant de recommencer l'algorithme de génération de colonnes.

Comme l'identification d'inégalités valides violées peut nécessiter la résolution d'un problème de séparation $\mathcal{NP}$-difficile, il est souvent nécessaire d'utiliser des heuristiques. Or, il est possible que ces heuristiques ne parviennent pas à trouver de plans coupants. Afin de contourner ces difficultés, on ajoute une ou plusieurs inégalités valides à la fin de l'algorithme de génération de colonnes uniquement lorsque la violation de la solution fractionnaire actuelle est suffisamment élevée par rapport à l'une des inégalités considérées.

Si la violation totale est trop faible, alors on procède à un branchement dichotomique. Ce processus consiste à créer deux nouveaux nœuds fils L_{a+1} et L_{a+2} dont le parent est l dans l'arbre de branchement à l'aide d'une décision tirée d'un ensemble préétabli $\mathcal{D}$. Dans chaque nœud, on restreint $S_{IP,k*}^l$, le domaine réalisable de PMR^l à la dernière itération de la génération de colonnes en fixant la valeur de certaines variables à des valeurs entières. On divise ce domaine en deux de manière à respecter $S_{IP,k*}^l = S_{IP,0}^{a+1} \cup S_{IP,0}^{a+2}$. On garantit ainsi l'optimalité de la solution, car on s'assure de ne pas éliminer de solution entière réalisable au cours de cette procédure.

On s'assure également de diviser le domaine réalisable du nœud père de telle sorte que $\underline{\lambda}^l_{k*} \notin S^{a+1}_{IP,0}$ ou $\underline{\lambda}^l_{k*} \notin S^{a+2}_{IP,0}$. La solution fractionnaire trouvée dans le nœud actuel devient alors irréalisable dans au moins un des deux nœuds enfants. Dans notre application particulière, on considère uniquement des règles qui rendent le domaine réalisable des nœuds fils disjoint, c.-à.-d. $S^{a+1}_{IP,0} \cap S^{a+2}_{IP,0} = \emptyset$. On partitionne ainsi $S^l_{IP,k*}$.

L'algorithme BCP se termine au noeud L_l lorsque la valeur de la meilleure solution entière est égale à la valeur de la solution de PMR^l_k, si le problème est non borné ou non réalisable, si on a exploré l'ensemble de l'arbre de branchement ou si on dépasse le temps maximal alloué de 20 minutes (1200 secondes). L'algorithme général est décrit ici et des détails sur chacune des sous-routines sont données dans les sections subséquentes.

Soit :
- $\mathscr{P}$: l'ensemble des routes réalisables
- $\mathscr{P}^l_k \subseteq \mathscr{P}$: l'ensemble restreint des routes réalisables considérées à l'itération k dans le l^e nœud de branchement

- $S^l_{IP,k} = \{\lambda_p : \sum_{p\in\mathscr{P}^l_k} \alpha_{ip}\lambda_p = 1 \ \forall i \in \mathcal{N}; \lambda_p \in \{0,1\} \forall p \in \mathscr{P}^l_k\}$: le domaine réalisable du problème maître restreint en nombres entiers au l^e nœud de branchement à la k^e itération de l'algorithme de génération de colonnes
- $S^l_{RP,k} = \{\lambda_p : \sum_{p\in\mathscr{P}^l_k} \alpha_{ip}\lambda_p = 1 \ \forall i \in \mathcal{N}; 0 \leq \lambda_p \leq 1 \forall p \in \mathscr{P}^l_k\}$: le domaine réalisable de la relaxation linéaire du problème maître restreint au l^e nœud de branchement à la k^e itération de l'algorithme de génération de colonnes
- $PMR^l_k = \{\min_\lambda \sum_{p\in\mathscr{P}^l_k} \hat{c}_p\lambda_p \ : \lambda \in S^l_{IP,k}\}$: le problème maître restreint à la k^e itération de l'algorithme de génération de colonnes au nœud de branchement l et considérant uniquement $\mathscr{P}^l_k$
- $R(PMR^l_k) = \{\min_\lambda \sum_{p\in\mathscr{P}^l_k} \hat{c}_p\lambda_p \ : \ \lambda \in S^l_{RP,k}\}$: la relaxation linéaire de PMR^l_k

- $\overline{\lambda}^l_k$: une solution entière de PMR^l_k
- $\underline{\lambda}^l_k$: la solution primale optimale de $R(PMR^l_k)$
- π^l_k : la solution duale calculée à partir de la solution optimale de $R(PMR)^l_k$
- $z(PMR)^l_k$: la valeur de λ^l_k
- $\overline{z}(PMR)^l_k$: une borne supérieure sur $z(PMR)^l_k$
- $\underline{z}(PMR)^l_k$: une borne inférieure sur $z(PMR)^l_k$

- SP_k^l : le sous-problème à l'itération k
- $z(SP)_k^l$: la valeur minimale de SP_k^l
- SP_{type} : le type de sous-problème utilisé - $SP_{type} \in \{$ elem, ng-paths, 2-cyc $\}$

- $A^0 = \{A_1, A_2, ... A_{|\mathcal{N}|}\}$: un ensemble de $|\mathcal{N}|$ colonnes de dimension $|\mathcal{N}|$ correspondant à des routes artificielles (o, i, o') $\forall i \in \mathcal{N}$ telles que $A_{ip} = 1$ si $p = i$ et $A_{ip} = 0$ sinon de coût $\hat{c}_p = M$ où M est un nombre réel positif très grand.
- λ^0 : les variables binaires correspondant à chacune des routes de A^0
- $\mathscr{P}^0$: les routes correspondant aux colonnes de A^0

- $S^0 = \{\lambda_p : \sum_{p \in \mathscr{P}^0} \alpha_{ip}\lambda_p = 1 \ \forall i \in \mathcal{N}; \lambda_p \in \{0, 1\} \forall p \in \mathscr{P}^0\}$: le domaine réalisable du problème maître restreint au 1$^{\text{er}}$ nœud de branchement à la 1$^{\text{ère}}$ itération de l'algorithme de génération de colonnes
- L : la liste des nœuds actifs dans l'arbre de branchement
- a : un compteur global de noeuds dans l'arbre de branchement
- $L_a \in L$: le a^e nœud de branchement actif dans l'arbre
- $p(L_a)$: le nœud parent du nœud L_a dans l'arbre de branchement
- $S_{d_1}^l = \{\lambda_p : \sum_{p \in \mathscr{P}_k^l} \tilde{\alpha}_{abp}^d \lambda_p < D_d, \ ; D_d \in \mathbb{Z}\}$: la restriction sur le domaine réalisable du problème maître imposé par la décision d dans le premier nœud de branchement. Cette décision rend la solution fractionnaire actuelle irréalisable dans au moins un des deux nœuds de branchement fils ($\tilde{\alpha}_{abp}^d$ représente la contribution d'un arc $(a, b) \in \mathcal{AS}$ dans la route p à cette contrainte).
- $S_{d_2}^l = \{\lambda_p : \sum_{p \in \mathscr{P}_k^l} \tilde{\alpha}_{abp}^d \lambda_p \geq D_d, \ ; D_d \in \mathbb{Z}\}$: la restriction sur le domaine réalisable du problème maître imposé par la décision d dans le deuxième nœud de branchement.
- $\mathcal{D}$: l'ensemble des décisions de branchement considérées à chaque nœud de branchement

- λ^* : la solution optimale de PM
- z^* : la valeur de λ^*

- C^l : l'ensemble des inégalités valides identifiées par l'algorithme de séparation au l^e nœud de branchement
- $viol^l$: la violation (différence entre le membre de droit et le membre de gauche) maximale de la solution fractionnaire actuelle par rapport aux inégalités valides considérées au l^e nœud de branchement
- $violSeuil$: un seuil critique pour la violation d'inégalités valides (fixe)
- $maxTemps$: temps maximal alloué pour la résolution totale du problème (fixe)

Algorithme 1: Branch-cut-and-price($\mathcal{GS}, SP_{type}$)

$a \leftarrow 0$
$L \leftarrow \{L_0\}$
$S^0_{IP,0} \leftarrow S^0, C^0 \leftarrow \emptyset$
$z^* \leftarrow \infty$, $\lambda^* \leftarrow \lambda^0$
tant que $L \neq \emptyset$ **ou** $temps > maxTemps$ **faire**
  $L_l \leftarrow$ **Choisit**(L)
  $L \leftarrow L \setminus L_l$
  $(\underline{\lambda}^l, \underline{z}(PMR)^l) \leftarrow$ **CalculBorneInf**$(\mathcal{GS}, S^l_{IP,0}, \mathscr{P}^l, SP_{type})$
  si $\underline{z}(PMR)^l = -\infty$ **alors retourne**
  sinon si $\underline{z}(PMR)^l = \infty$ **alors continue**
  sinon si $\underline{\lambda}^l \in \mathbb{N}$ **alors**
    $\overline{\lambda}^l \leftarrow \underline{\lambda}^l$
    si $\overline{z}(PMR)^l < z^*$ **alors**
      $\lambda^* = \underline{\lambda}^l$
      $z^* \leftarrow \overline{z}(PMR)^l$
      $L \leftarrow L \setminus \{L_s : \underline{z}(PMR)^{p(L_s)} \geq z^*\}$
    fin
  sinon
    $(viol^l, C^l) \leftarrow$ **TrouverCoupesViolées**$(\underline{\lambda}^l, S^l_{IP,k*})$
    si $viol^l \geq violTreshold$ **alors**
      $S^{a+1}_{IP,0} \leftarrow S^l_{IP,k*} \cap C^l$
      $p(L_{a+1}) \leftarrow L_l$
      $L \leftarrow L \cup \{L_{a+1}\}$
      $a \leftarrow a + 1$
    sinon
      $d \leftarrow$**EvaluerDecisionsBranchement**$(\underline{\lambda}^l, S^l_{IP,k*}, \mathcal{D})$
      $S^{a+1}_{IP,0} \leftarrow S^l_{IP,k*} \cap S^l_{d_1}, S^{a+2}_{IP,0} \leftarrow S^l_{IP,k*} \cap S^l_{d_2}$
      $p(L_{a+2}) \leftarrow p(L_{a+1}) \leftarrow L_l$
      $L \leftarrow L \cup \{L_{a+1}, L_{a+2}\}$
      $a \leftarrow a + 2$
    fin
  fin
fin
retourne (λ^*, z^*)

La méthode **Choisit** sélectionne le meilleur nœud de branchement à considérer selon un certain critère. Dans notre algorithme, nous adoptons une méthode « best first » qui consiste à identifier le nœud dont le nœud père a une borne inférieure minimale dans une branche non élaguée.

CalculBorneInf est l'algorithme de génération de colonnes qui procure la borne inférieure sur la valeur de la solution optimale au problème que l'on cherche à résoudre à ce nœud de branchement. Cette borne inférieure sur PMR_k^l est obtenue grâce à l'algorithme 2 décrit dans les pages suivantes. Une itération de l'algorithme consiste donc en : 1) Une résolution de le relaxation linéaire du problème maître menant à l'obtention d'une solution primale optimale ainsi qu'une solution duale associée ; 2) L'identification de colonnes de coût réduit négatif, si ces dernières existent. L'identification de colonnes peut être faite à l'aide d'un algorithme exact ou d'une heuristique.

La condition à la ligne 9 est respectée si la relaxation linéaire de PMR_k^l est non bornée. Comme les variables associées au choix des routes et les coûts $\hat{c}_p$ sont bornés pour tout PMR_k^l, cette condition sera toujours fausse. La condition à la ligne 10 est respectée lorsque la relaxation de PMR_k^l est non réalisable. La ligne 16 élague les branches de l'arbre de branchement inutiles.

TrouverCoupesViolées identifie les inégalités valides violées par la solution fractionnaire actuelle à la dernière itération de l'algorithme de génération de colonnes au nœud L_l. On utilise la librairie CVRPSEP (Lysgaard., 2004) pour identifier certaines des ces inégalités. On utilise également une routine d'énumération explicite pour identifier un autre type d'inégalité (voir la section 4.4.1). Tous les algorithmes de séparation sont exécutés sur le graphe $\mathcal{G}$ original. Pour passer d'une solution dans $\mathcal{GS}$ à une solution dans $\mathcal{G}$, on agrège tous les nœuds associés à un client en un seul nœud, c.-à.-d. les nœuds $\mathscr{V}(\mu, i) \in C(i)$ où $\mu \in \{1, 2, ..., Q - E[\xi_i]\}$ associés à un client i donné et on accumule le flot sur les arcs entrant et sortant de ce nœud agrégé (correspondant à l'ensemble $C(i)$).

Si l'algorithme de séparation ne parvient pas à trouver d'inégalités valides suffisamment violées, alors on effectue un branchement dichotomique. **EvaluerDecisionsBranchement** identifie la meilleure règle de branchement selon la violation de la solution fractionnaire en cours $\underline{\lambda}^l$ par rapport au domaine réalisable $S^l_{IP,k*}$. Cette règle est tirée de $\mathcal{D}$, l'ensemble des règles de branchement fixées préalablement et disposant d'une mesure de la qualité des décisions par rapport à la violation actuelle. Comme on le voit aux lignes 22 et 28, on crée un seul nœud de branchement lorsqu'on utilise les inégalités valides pour obtenir une solution entière, mais on crée deux nœuds lorsqu'on utilise une règle de branchement dichotomique. La décision est appliquée à la ligne 21. Le domaine du problème maître restreint sera modifié dans tous les nœuds fils si cette décision est imposée au niveau du problème maître. Sinon, le graphe résiduel $\mathcal{GS}$ sera modifié dans ces nœuds de branchement.

L'algorithme **Branch-cut-and-price** parvient à trouver la solution optimale si $L = \emptyset$ et $z^* < \infty$ (il n'y a plus de noeuds de branchement à évaluer et une solution entière réalisable a été trouvée). La procédure retourne alors λ^*, la solution optimale.

Algorithme 2: CalculBorneInf($\mathcal{GS}, S^l_{IP,0}, \mathscr{P}^l, SP_{type}$)

```
1  k ← 0
2  répéter
3      pour chaque Algo ∈ { RechercheTaboue, (E)SPPRC } faire
4          répéter
5              (λ̲^l_k, π_k, z̲(PMR^l_k)) ← Résoudre (R(PMR^l_k), S^l_{RP,k})
6              P^l_k ← P^l_k ∪ Algo (GS, S^l_{IP,k}, λ̲^l_k, π_k)
7              z(SP^l_k) ← min_{p∈P^l_k} {c̄_p}
8              k ← k + 1
9              P^l_k ← P^l_{k−1}
10         tant que z(SP^l_{k−1}) < 0 et Algo=RechercheTaboue;
11     fin
12 tant que z(SP^l_{k−1}) < 0;
13 retourne(λ̲^l_k, z̲(PMR^l_k))
```

Résoudre est la méthode qui fait appel à un solveur linéaire comme `CPLEX` pour obtenir la solution optimale à $R(PMR^l_k)$, la relaxation linéaire du problème maître restreint de partitionnement d'ensemble avec le domaine réalisable $S^l_{RP,k}$ en considérant $\mathscr{P}^l_k$, l'ensemble des routes considérées à cette itération.

Algo correspond à la résolution de l'algorithme heuristique de recherche tabou (**RechercheTaboue**) ou de l'algorithme exact de résolution du problème de plus court chemin (**(E)SPPRC**). À chaque itération de l'algorithme de génération de colonnes (chaque itération de la boucle 2 à 12), on considère d'abord l'algorithme heuristique et ensuite l'algorithme exact.

La boucle 4 à 10 assure qu'à chaque itération de l'algorithme de génération de colonnes on utilise l'algorithme tabou jusqu'à ce qu'il ne soit plus possible d'identifier de colonnes de coût réduit négatif c.-à-d. lorsque le plus petit coût réduit trouvé par cet algorithme à l'itération k et identifié à la ligne 7 soit plus grand ou égal à zéro. Lorsque la méthode taboue

échoue à identifier des colonnes de coût réduit négatif, on passe à la méthode exacte. Si cette dernière parvient à identifier des colonnes améliorant l'objectif, on entre dans une nouvelle itération de l'algorithme de génération de colonnes (on passe à la ligne 2). Si on ne trouve plus de colonnes de coût réduit négatif, l'algorithme de génération de colonnes se termine et la solution de la relaxation linéaire à ce noeud de branchement est optimale. On exécute donc *au plus* 1 fois l'algorithme d'étiquetage exact à chaque itération de la boucle 4 à 10 et *exactement* une fois à chaque itération de la boucle 2 à 12, ce qui est susceptible de limiter le temps total de résolution.

RechercheTaboue correspond à une heuristique qui permet de générer rapidement des colonnes de coût réduit négatif sans avoir à résoudre un problème de plus court chemin avec contraintes de ressources. Cet algorithme est décrit dans la section 4.3. Comme cet algorithme est heuristique, il est possible qu'il ne parvienne pas à trouver de colonnes de coût réduit négatif, mais qu'il en existe quand même.

(E)SPPRC correspond à la résolution d'un problème de plus court chemin avec contraintes de ressources (avec élimination de 2-cycles, avec voisinage ou élémentaire) à l'aide d'un algorithme d'étiquetage. Cette procédure est décrite dans la section 4.2 suivante. Comme l'algorithme (E)SPPRC est exact, si ce dernier ne parvient pas à identifier de colonnes de coût réduit négatif, alors on sait qu'il n'en existe plus à générer à ce noeud de branchement et que la relaxation linéaire du problème maître restreint est optimale. La procédure **CalculBorneInf** se termine alors.

On note qu'il est possible que ces deux méthodes ajoutent plus d'une route de coût réduit négatif à l'ensemble des routes considérées à chaque itération k. De plus, lorsque des routes de coût réduit négatif sont identifiées, elles sont ajoutées au problème maître.

4.2 Résolution du sous-problème

4.2.1 Formulation générique du SPPRC

Lorsqu'on n'impose pas de contraintes sur la présence de cycles, la résolution du sous-problème est équivalente à la résolution d'un problème de plus court chemin avec contraintes de ressources sur le graphe $\mathcal{G}$ (« shortest path problem with resource constraints » - SPPRC). Ce problème peut s'énoncer comme suit : étant donné un réseau avec une source et un puits composé d'un ensemble de nœuds reliés entre eux par des arcs auxquels sont associés

des coûts ainsi qu'un ensemble de ressources dont la consommation est définie sur les arcs et dont la consommation totale est limitée sur chacun des nœuds, quel est le chemin de la source au puits ayant un coût minimal ? Ce type de problème est essentiel pour la résolution de problèmes de tournées de véhicules par génération de colonnes, car il permet d'identifier les routes de coût réduit strictement négatif pouvant améliorer la solution actuelle du problème maître restreint (Irnich et Desaulniers, 2005).

Sous sa forme la plus simple, une ressource est une quantité que l'on accumule le long des arcs traversés par un chemin dans le réseau considéré. Par exemple, les ressources utilisées dans un VRPTW classique incluent la quantité totale livrée ou cueillie ainsi que le temps cumulé le long d'un chemin alors que pour le VRP, on considère uniquement la quantité totale livrée ou cueillie. Le coût total est une ressource, car il s'agit d'un quantité qu'on accumule sur chaque arc. Toutefois, on le traite souvent séparément, car il est généralement non borné et possède une importance particulière. On considère $\mathcal{RES}$ comme l'ensemble fini des $R = |\mathcal{RES}|$ ressources considérées dont la consommation minimale sur chaque arc (u, v) est donnée par τ^s_{uv} $\forall s \in \mathcal{RES}$. On pose également $\mathbf{r_u} \in \mathbb{R}^R$ comme le vecteur de consommation de ressource indiquant la consommation totale au nœud u.

La consommation totale de chacune de ces ressources le long d'un chemin p_u se terminant au nœud u, ayant un vecteur de consommation $\mathbf{r_u}$ et empruntant l'arc (u, v) est déterminée en partie par un vecteur de fonctions d'extension $f_{uv} = (f^s_{uv})^{|\mathcal{RES}|}_{s=1}$. La consommation totale d'une ressource $s \in \mathcal{RES}$ est donnée sous a forme générique par la fonction d'extension $f^s_{uv}(\mathbf{r_u}) : \mathbb{R}^{|\mathcal{RES}|} \to \mathbb{R}$; ce qui signifiqiue que les fonctions d'extensions de chaque ressource peuvent dépendre du vecteur de consommation en entier. Ces fonctions procurent une borne inférieure sur la valeur que peut prendre la consommation totale de chacune des ressources au nœud v après avoir visité l'arc (u, v). On a donc $f^s_{uv}(\mathbf{r_u}) \leq r^s_v$ $\forall s \in \mathcal{RES}$.

La consommation totale de chaque ressource est également bornée sur chacun des nœuds. Ces contraintes prennent la forme de fenêtres de consommation de ressources définies par une borne inférieure et supérieure de la forme $[a^s_v, b^s_v]$ où $a^s_v \in \mathbb{N}$ et $b^s_v \in \mathbb{N}$ pour chaque ressource s et chaque nœud v. Dans la plupart des cas, on considère des bornes fortes et la quantité consommée au total à chaque nœud doit tomber directement dans l'intervalle définie. Si la consommation est inférieure à a^s_v, alors on la fixe à a^s_v. Ceci permet notamment de modéliser l'attente si la ressource considérée est le temps cumulé et que les livraisons doivent avoir lieu dans un intervalle spécifique. Plus spécifiquement, la consommation totale de ressources est limitée sur chaque nœud v par $\tilde{a}^s_v \leq r^s_v \leq b^s_v$ où $\tilde{a}_v = \max_{u:(u,v)\in\delta^-(\{v\})}\{f^s_{uv}(\mathbf{r_u}), a^s_v\}$.

4.2.2 Formulation spécifique du SPPRC pour notre problème

Dans notre problème particulier, nous tentons de résoudre un SPPRC sur le graphe initial $\mathcal{G}$. En effet, tel que décrit à la section 3.4, le problème DEP que nous essayons initialement de résoudre impose que l'on identifie des tournées dont la demande cumulée espérée ne dépasse pas la capacité des véhicules. Toutefois, comme le calcul du coût espéré d'échec est facilité par l'utilisation du graphe espace-état $\mathcal{GS}$, on utilise plutôt ce graphe lors de la résolution de notre sous-problème.

La formulation $DEPs$ ne considère pas explicitement de ressources autres que $\bar{c}$. La demande cumulée (espérée) est effectivement ajoutée explicitement à chaque nœud du graphe $\mathcal{GS}$ et il n'est pas nécessaire d'ajouter cette ressource additionnelle à cette formulation. On considère toutefois dem, la demande cumulée espérée comme une ressource. Comme on le verra plus tard, il est utile de prendre en compte explicitement cette ressource, car elle est essentielle pour l'algorithme d'étiquetage bidirectionnel.

On considère donc initialement $\mathcal{RES} = \{\bar{c}, \text{dem}\}$, un ensemble de $|\mathcal{RES}| = R = 2$ ressources. Le coût d'un arc $(u, v) \in \mathcal{AS}$ est donné par $\bar{c}_{uv}$, le coût réduit qu'on calcule à partir de la valeur des variables duales associées aux contraintes du problème maître restreint à l'itération précédente de l'algorithme de génération de colonnes.

On ne considère que les chemins réalisables en moyenne, c'est-à-dire ceux de demande cumulée espérée inférieure ou égale à la capacité d'un véhicule. Par ailleurs, chaque client a une demande espérée entière et strictement positive. On a donc $a_u^{\text{dem}} = 1$ et $b_u^{\text{dem}} = Q,\ \forall u \in \mathcal{NS}$. Comme on utilise le graphe $\mathcal{GS}$ et qu'il est uniquement possible d'arriver à un nœud $u = \mathscr{V}(\mu, i)$ avec une demande cumulée de μ, on peut resserrer cet intervalle à $[\mu, \mu]\ \forall\ u = \mathscr{V}(\mu, i) \in \mathcal{NS}$. La consommation minimale (et maximale) de cette ressource sur chaque arc $(u, v) \in \mathcal{AS}$ est donnée par $\tau_{uv}^{\text{dem}} = E[\xi_j]$ si $v \in C(j)$. On note que $\bar{c}_p$, le coût réduit d'un chemin p est considéré comme une ressource, que sa consommation totale n'est pas bornée ($b_u^{\bar{c}} = \infty\ \forall u \in \mathcal{NS}$) et que sa consommation minimale (et maximale) sur chaque arc est donnée par $\tau_{uv}^{\bar{c}} = \bar{c}_{uv}$.

Dans notre problème, on suppose que les fonctions d'extension $f_{uv}^s(\mathbf{r_u})$ pour le coût réduit et la demande cumulée sont décomposables par ressources. Elles prennent la forme très simple

$f^s_{uv}(\mathbf{r_u}) = r^s_u + \tau^s_{uv}$. On obtient donc $f^{\bar{c}}_{uv}(\mathbf{r_u}) = r^{\bar{c}}_u + \bar{c}_{uv}$ et $f^s_{uv}(\mathbf{r_u}) = r^{\text{dem}}_u + E[\xi_j]$. Le coût réduit total d'une route est donc donné par la somme des coûts réduits des arcs empruntés et la demande cumulée espérée est donnée par la somme des demandes espérées des clients visités par cette route.

Afin de bien comprendre la complexité du SPPRC et l'importance de l'ajout des contraintes de ressources, il est intéressant de le comparer à une de ses spécialisation bien connues : le problème de plus court chemin (« shortest path problem » - SPP).

4.2.3 Différence entre le SPP et le SPPRC

Bien que le SPPRC ressemble beaucoup au problème de plus court chemin sans restriction sur la consommation de ressources, ces deux problèmes possèdent des différences fondamentales. On note immédiatement qu'il est impossible d'appliquer les algorithmes standard de résolution de SPP comme Dijsktra et Bellman-Ford pour résoudre un SPPRC *sur le graphe* $\mathcal{G}$ *original*. En effet, l'algorithme de Dijsktra ne peut gérer les arcs de coût négatif alors que celui de Bellman-Ford ne peut se compléter s'il existe des cycles de coût négatif. Or, dans un contexte de génération de colonnes, on considère la valeur des variables duales et ces dernières peuvent prendre des valeurs négatives. Il y a donc de fortes chances que l'algorithme de plus court chemin emprunte des arcs et des cycles de coût réduit négatif. Contrairement au SPP, il est toutefois possible de trouver des solutions optimales au SPPRC lorsqu'il existe des cycles négatifs, car les contraintes de ressources empêchent de cycler infiniment. Il sera effectivement impossible de cycler sans fin s'il existe une ressource bornée de consommation minimale strictement positive sur tous les arcs : $\exists s \ : \ \tau^s_{uv} > 0 \wedge b^s_v < \infty \ \forall (u,v) \in \mathcal{AS}$.

Il est également impossible d'appliquer ces deux algorithme standards pour résoudre un SPPRC *sur* $\mathcal{G}$, car ils ne prennent pas en compte la faisabilité d'un chemin par rapport aux contraintes de ressources et considèrent uniquement le coût des chemins. Toutefois, le SPPRC représente un problème de décision multi-critère, car il considère un vecteur de consommation de ressources plutôt qu'un seul scalaire (le coût d'un chemin). Ainsi, lors de sa résolution, on ne peut se contenter de conserver uniquement le chemin de plus petit coût à un nœud donné. On doit plutôt prendre en compte tous les chemins qui pourraient mener à un chemin optimal de la source au puits considérant la restriction sur les ressources.

La résolution du SPPRC présente des difficultés théoriques additionnelles par rapport au SPP. Le SPP peut être résolu efficacement en temps polynomial, car ce problème respecte le principe d'optimalité de Bellman qui peut être décrit comme suit (Bertsekas, 2005). On

considère un problème de plus court chemin dans le réseau original $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ avec une source o et un puits o', $n+2$ nœuds ($\mathcal{N} = \{1, 2, ..., n\} \cup \{o, o'\}$) et m arcs $(i, j) \in \mathcal{A}$ de coût $\bar{c}_{ij}$. On suppose également que $\mathcal{L}$ représente le nombre maximal et fini d'étapes nécessaires à la résolution du problème et que p^* représente un chemin réalisable de la source au puits de coût minimal $\bar{c}_{p^*} = \sum_{(i,j)\in p^*} \bar{c}_{ij}$. On pose ensuite $\pi = \{\mu_1, \mu_2, ..., \mu_{\mathcal{L}-1}\}$ comme une politique comprenant un ensemble de décisions μ_t correspondant au choix d'un arc à suivre à l'étape $t \in \{1, 2, ..., \mathcal{L} - 1\}$. À chaque étape t est associé un état $x_t = i$ qui indique le nœud final du chemin partiel de o à i dans $\mathcal{G}$ après avoir parcouru $t - 1$ arcs.

Le principe de Bellman stipule qu'étant donné une politique optimale $\pi^* = \{\mu_1^*, \mu_2^*, ..., \mu_{\mathcal{L}-1}^*\}$ qui minimise $\bar{c}_p$, le coût réduit total de déplacement d'un chemin de la source au puits et qui mène au nœud i à l'étape t, alors la politique tronquée $\{\mu_t^*, \mu_{t+1}^*, ..., \mu_{\mathcal{L}-1}^*\}$ est encore optimale pour le sous-problème consistant à déterminer le plus court chemin du nœud i au puits. Le SPP possède donc une sous-structure optimale et l'optimisation locale (le choix de l'arc (i, j) de plus petit coût $\bar{c}_{ij}$ sortant du nœud final actuel i) à chaque étape permet d'optimiser le problème en entier. Ceci explique pourquoi le SPP se prête bien à la programmation dynamique et peut être résolu en temps polynomial par des algorithmes standards comme celui de Dijsktra et de Bellman-Ford.

Cependant, le SPPRC ne respecte pas ce principe sous cette forme et il est nécessaire de l'étendre pour l'appliquer à ce problème. Soit $x_t = i$ un état réalisable à l'étape t et $\mathbf{r}(x_t)$, un vecteur représentant la consommation de ressources le long du chemin partiel de la source au nœud $i \in \mathcal{N}$. On définit également l'ensemble des vecteurs de ressources Pareto optimaux à l'étape t pour l'état x_t comme les vecteurs $\mathbf{r}$ tels qu'il n'existe pas d'autres vecteurs $\mathbf{r}'$ où $r'^s(x_t) \leq r^s(x_t)\ \forall s \in \mathcal{RES}$ et au moins une de ces inégalités est stricte.

Une politique optimale $\pi'^* = \{\mu_1'^*, \mu_2'^*, ..., \mu_{\mathcal{L}-1}'^*\}$ conserve donc uniquement les vecteurs de ressources Pareto optimaux pour chaque état x_t à chaque étape t [1]. Si on considère une telle politique optimale et que cette dernière nous amène au nœud i à l'étape t, alors la politique tronquée $\{\mu_t'^*, \mu_{t+1}'^*, ..., \mu_{\mathcal{L}-1}'^*\}$ est encore optimale pour le sous-problème consistant à déterminer l'ensemble des vecteurs Pareto optimaux associés à la consommation de ressources d'un chemin de i jusqu'au puits.

1. La politique optimale $\pi^* = \{\mu_1^*, \mu_2^*, ..., \mu_{\mathcal{L}-1}^*\}$ pour le SPP comprend un maximum de $n+1$ étapes si le réseau comprend $n+2$ nœuds. Ceci est dû au fait que le graphe sous-jacent ne peut contenir de cycles négatifs si une solution optimale de coût fini existe. Comme il n'existe pas de cycles négatifs, il ne peut être optimal pour un chemin partiel minimal de revisiter un nœud. Il s'ensuit que dans le cas nécessitant le plus d'étapes,

Si on considère $\mathcal{F}_{\mathcal{G}}(i,j)$ comme l'ensemble des chemins de i à j dans $\mathcal{G}$ réalisables par rapport à la consommation de ressources, $\mathcal{S}_{\mathcal{G}}$ comme l'ensemble des chemins réalisables par rapport à l'élimination des cycles et $\mathcal{T}_{\mathcal{G}}(p)$ l'ensemble des vecteurs de ressource réalisables pour le chemin p, alors le SPPRC exécuté sur le graphe $\mathcal{G}$ original peut être formulé comme :

$$\min_{p \in \mathcal{F}_{\mathcal{G}}(o,o') \cap \mathcal{S}_{\mathcal{G}}} \left(\min_{\mathbf{r} \in \mathcal{T}_{\mathcal{G}}(p)} \mathbf{r} \right) \tag{4.1}$$

où la minimisation $\min_{\mathbf{r} \in \mathcal{T}_{\mathcal{G}}(p)}$ indique que l'on conserve les vecteurs de ressources Pareto optimaux et $\mathcal{F}_{\mathcal{G}}(o,o') = \{$ toutes les routes réalisables de o à $o'\}$, $\mathcal{S}_{\mathcal{G}} = \{$ toutes les routes $\}$ si on considère le SPPRC. À l'opposé, la résolution d'un SPP correspond à la résolution du problème :

$$\min_{p \in \mathcal{F}_{\mathcal{G}}(o,o') \cap \mathcal{S}_{\mathcal{G}}} \left(\min_{r^{\bar{c}}: \mathbf{r} \in \mathcal{T}_{\mathcal{G}}(p)} \mathbf{r} \right) \tag{4.2}$$

où $\mathcal{F}_{\mathcal{G}}(o,o') = \{$toutes les routes de o à $o'\}$, $\mathcal{S}_{\mathcal{G}} = \{$toutes les routes$\}$ pour le SPP et la minimisation $\min_{r^{\bar{c}}: \mathbf{r} \in \mathcal{T}_{\mathcal{G}}(p)}$ implique que l'on considère uniquement le coût réduit.

En somme, ces deux problèmes possèdent une structure très similaire et il est possible de leur appliquer le principe de Bellman. La complexité de l'algorithme de résolution du SPPRC est néanmoins augmentée par rapport aux algorithmes pour SPP comme celui de Bellman-Ford et de Dijkstra, car on doit maintenant maintenir un ensemble de vecteurs de ressources à chaque nœud plutôt qu'uniquement le meilleur coût comme c'était le cas pour le SPP. La complexité dépend donc fortement du nombre de vecteurs Pareto optimaux traités à chaque nœud ainsi que du nombre de ressources considérées. On note que le nombre total d'étiquettes correspondant à des chemins partiels réalisables de l'origine à un nœud donné augmente rapidement selon le nombre de critères (ressources) considérés ainsi que selon la taille du chemin partiel. Lorsque l'on considère uniquement la demande cumulée espérée et que l'on permet la présence de chemins avec cycles, il existe un maximum de $Q - E[\xi]$ « états » à chaque client $i \in \mathcal{N}$ et donc $O(nQ)$ états au total. Ceci explique en partie pourquoi le SPPRC est

le plus court chemin sera formé par un chemin Hamiltonien de coût minimum contenant exactement $n+1$ arcs.

Par contre, la politique optimale π^* pour le SPPRC peut comprendre plus de $n+1$ étapes lorsque le graphe contient des cycles de coût négatif, car dans ce cas, il peut être optimal de visiter un nœud plus d'une fois. On est toutefois assuré qu'on sera capable de résoudre ce problème en un nombre fini d'étapes si la consommation d'au moins une ressource est strictement positive sur chaque arc et que la fenêtre de consommation pour cette ressource est bornée.

$\mathcal{NP}$-difficile (Garey et Johnson, 1979).

Les travaux de (Desrochers et Soumis, 1988) et (Desrochers *et al.*, 1992) démontrent toutefois qu'il est possible de le résoudre à l'aide d'un algorithme pseudo-polynomial de complexité $O(D^2)$ où D représente le nombre maximal d'états pour tous les nœuds de $\mathcal{N}$. Il est ainsi possible de résoudre le SPPRC sur le graphe $\mathcal{G}$ avec un algorithme pseudo-polynomial de complexité $O((nQ)^2)$. Bien que la capacité puisse être relativement grande par rapport au nombre de clients dans certaines instances, certaines instances peuvent être résolues en temps « raisonnable » .

On note qu'on considère implicitement la contrainte $\sum_{(i,j)\in p} E[\xi_j]x_{ij}^v \leq Q,\ \forall v \in \mathcal{V}$ (imposant que la demande espérée cumulée le long de la route p empruntée par le véhicule v ne dépasse pas la capacité du véhicule) grâce à la discrétisation des demandes cumulées et l'utilisation des nœuds de $\mathcal{NS}$. Comme le graphe $\mathcal{GS}$ considère la faisabilité des routes, un algorithme qui minimise le coût total des routes dans le graphe espace-état trouvera le même chemin que celui de coût minimal trouvé par un algorithme considérant les vecteurs Pareto optimaux composés des ressources de capacité et de coût dans le graphe $\mathcal{G}$. Le SPPRC sur le graphe $\mathcal{G}$ est donc équivalent à un SPP sur le graphe $\mathcal{GS}$ *si on considère uniquement 2 ressources* (le coût et la capacité/demande cumulée). Par exemple, si on considérait également le temps cumulé, on ne pourrait se contenter de résoudre un SPP sur $\mathcal{G}$ pour obtenir la même solution que celle obtenue en résolvant un SPPRC sur $\mathcal{GS}$.

On observe également que le graphe $\mathcal{GS}$ est acyclique, mais qu'il peut contenir des arcs de coût réduit négatifs. On ne peut donc pas utiliser l'algorithme de Disjktra. Toutefois, si on utilise un algorithme comme celui de Bellman-Ford dont la complexité est de l'ordre de $O(|\mathcal{NS}| + |\mathcal{AS}|)$ pour les graphes acycliques (on parcourt la liste d'adjacence du graphe), alors on obtient un algorithme de complexité $O(nQ + n^2Q) \in O(n^2Q)$.

Cette complexité est inférieure à celle de l'algorithme de (Desrochers *et al.*, 1992) sur $\mathcal{G}$. Cette observation peut être expliquée par le constat suivant. Lorsqu'on considère *deux* ressources et qu'on exécute l'algorithme d'étiquetage de (Desrochers *et al.*, 1992) sur le graphe $\mathcal{G}$, il peut y exister $O(Q)$ états Pareto optimaux pour chaque *client*. Ceci peut notamment être le cas si pour toutes les paires d'étiquettes L_i^1, L_i^2 arrivant à tout client $i \in \mathcal{N}$ donné, on a $\bar{c}(L_i^1) < \bar{c}(L_i^2) \wedge r^{\text{dem}}(L_i^1) > r^{\text{dem}}(L_i^2))$. Ces $O(Q)$ étiquettes Pareto optimales (permanentes) peuvent engendrer $O(n)$ nouvelles étiquettes et chacune de ces nouvelles étiquettes doivent être comparées à $O(Q)$ étiquettes existants afin de déterminer si elles sont Pareto optimales

(étape de dominance). Comme on doit répéter cette étape d'extension et de dominance pour tous les clients, on obtient bien la complexité théorique de $O((nQ)^2)$. On observe que le facteur de Q additionnel par rapport à l'algorithme de Bellman-Ford sur $\mathcal{GS}$ est attribuable à l'étape de dominance. Théoriquement, il serait donc préférable d'utiliser un algorithme de plus court chemin sur le graphe $\mathcal{GS}$. Toutefois, un tel algorithme sera exact uniquement si l'on considère deux ressources. Or, comme on le verra plus tard, l'ajout de ressources additionnelles est essentiel pour empêcher le retour à certains clients.

Par contre, si on exécute l'algorithme de (Desrochers *et al.*, 1992) sur le graphe $\mathcal{GS}$ et que l'on considère uniquement le coût réduit comme critère de dominance, alors on obtient un algorithme exactement équivalent à l'algorithme de Bellman-Ford sur le graphe $\mathcal{GS}$. En effet, dans ce cas, on effectue la dominance et l'extension vers chacun des noeuds plutôt que vers les clients comme c'était le cas pour l'algorithme d'étiquetage sur $\mathcal{G}$. Il peut y exister uniquement une étiquette Pareto optimale à chaque noeud $u = \mathscr{V}(\mu, i) \in \mathcal{NS}$, car $r^{\text{dem}}(L_u) = \mu;\ \forall L_u$ (chaque étiquette associée au noeud u a forcément une demande cumulée de μ et il suffit donc de conserver uniquement celle de meilleur coût réduit). Ainsi, pour chaque noeud dans $\mathcal{NS}$, on peut étendre l'étiquette Pareto optimale vers $O(n)$ noeuds (il existe uniquement des arcs entre $\mathscr{V}(\mu, i)$ et $\mathscr{V}(\mu + E[\xi_j], j)$ pour tous les clients $j \in \mathcal{N} \setminus \{i\}$ si $\mu + E[\xi_j] \leq Q$). Comme il peut uniquement y exister une étiquette Pareto optimale à chaque noeud, la dominance pour chaque nouvelle étiquette se fait en temps constant (on observe que cette dominance est exactement identique à la relaxation des arrêtes dans l'algorithme de Bellman-Ford). On obtient ainsi un algorithme de complexité $O(n^2Q)$.

Appliquer l'algorithme de Bellman-Ford sur le graphe $\mathcal{GS}$ correspond donc à résoudre un SPPRC sur le graphe $\mathcal{G}$ où l'on n'empêche aucun cycle, où l'on considère uniquement le coût réduit et la demande cumulée comme ressources et où l'on conserve tous les chemins partiels de plus petit coût arrivant à un client donné si ces chemins ont une demande cumulée différente. On effectue donc uniquement une dominance partielle. La dominance complète (ou agrégée, car elle considère tous les noeuds associés à un client donnée dans $C(i) \subset \mathcal{NS}\ \forall i \in \mathcal{N}$ lors de la dominance) correspond au SPPRC sur $\mathcal{G}$ et donne un algorithme de complexité $O((nQ)^2)$. Comme on l'explique plus tard, la performance empirique de l'algorithme avec dominance agrégée se révèle cependant plus rapide que celui sans cette dernière. Ce facteur semble surtout attribuable à la capacité de l'algorithme de dominer les étiquettes avec des demandes cumulées inférieures à celles du noeud de $\mathcal{NS}$ actuellement considéré.

4.2.4 Algorithme d'extension d'étiquettes générique pour le SPPRC

Tel qu'expliqué à la section précédente, on doit utiliser un algorithme plus complexe que ceux utilisés pour résoudre le SPP afin de résoudre le SPPRC. Il est notamment possible d'appliquer un algorithme de programmation dynamique avec fixation d'étiquettes qui considère les contraintes de ressources. Notre méthode se base sur les travaux séminaux de (Desrochers *et al.*, 1992) et emprunte plusieurs notations utilisées entres autres par (Feillet *et al.*, 2004).

On applique l'algorithme sur le graphe $\mathcal{GS}$. Comme pour les algorithmes d'étiquetage standard, chaque étiquette L est associée à un chemin partiel d'une origine unique à un nœud donné et possède un vecteur de consommation de ressources $\mathbf{r}(L)$ [2]. Dans notre algorithme de base, on considère dem, la demande cumulée espérée ainsi que le coût réduit comme des ressources. Cependant, lorsqu'on utilise les ng-routes ou le problème élémentaire et qu'on doit éliminer des cycles de taille $\kappa > 2$, on utilise également des ressources de visite binaires indiquant si un client j peut être visité par une route ($r^{\text{visit}_j}(L)\ \forall j \in \mathcal{N}$).

L'algorithme débute à partir d'une origine unique avec une consommation de ressource nulle pour toutes les ressources. Il tente ensuite de faire progresser les chemins partiels réalisables vers le puits en traitant les nœuds de manière répétée. À chaque itération, on considère un nœud et on « tire » les étiquettes associées aux nœuds prédécesseurs. Cette extension implique la mise à jour du vecteur de ressources selon la consommation de ces dernières sur les arcs empruntés. On arrête ce processus lorsqu'on a considéré tous les chemins utiles et réalisables de l'origine à la destination. On applique ensuite une procédure de filtrage pour identifier ceux de coût réduit négatif.

L'efficacité et la complexité de tout algorithme d'étiquetage dépendent de sa capacité à réduire le nombre d'étiquettes considérées à chaque itération. La plupart des algorithmes utilisent donc une règle de dominance afin de conserver uniquement les étiquettes qui permettent d'obtenir des chemins optimaux à un nœud donné. On applique cette règle après avoir terminé l'extension des étiquettes vers un nœud donné et on élimine les étiquettes inutiles.

Les sections suivantes expliquent les améliorations apportées à l'algorithme de base afin de limiter le nombre d'étiquettes considérées tout en garantissant l'optimalité de la solution. On explique également comment on peut généraliser le SPPRC pour produire des chemins

2. On utilise sans distinction les termes « étiquette » et « chemin », car chaque chemin partiel dans $\mathcal{GS}$ considéré par l'algorithme peut être associé à un chemin unique dans $\mathcal{G}$ ainsi qu'une étiquette unique.

élémentaires et tenir en compte les contraintes de partitionnement du problème maître.

4.2.5 Modification pour prendre en compte l'extension bidirectionnelle

Il est possible d'accélérer l'exécution de l'algorithme générique en utilisant un algorithme bidirectionnel. On peut effectivement utiliser un algorithme d'étiquetage « avant » qui progresse de la source os vers le puits os' et une procédure « arrière » qui avance du puits vers la source dans $\mathcal{GS}$. On arrête la progression de chaque algorithme à un nœud $u \in \mathcal{NS}$ se situant à « mi-chemin » et on peut ensuite fusionner les chemins partiels $os - u$ aux chemins $os' - u$.

On débute l'algorithme « avant » en fixant une étiquette L_{os} au nœud source os avec une consommation de ressource nulle pour chacune des ressources et un coût réduit nul. On fait ensuite progresser itérativement cette étiquette vers le puits en augmentant la consommation des ressources à chaque fois qu'on la tire vers un nœud successeur. On étend l'étiquette L_u basée au nœud prédécesseur $u = \mathscr{V}(\mu_i, i) \in C(i), i \in \mathcal{N}$ à l'aide d'un arc $(u, v) \in \mathcal{AS}$ vers le nœud atteignable présentement considéré $v = \mathscr{V}(\mu_j, j) \in C(j), j \in \mathcal{N}$ de manière à respecter les fenêtres de ressource $[a_v^s, b_v^s]$ pour toutes les ressources $s \in \mathcal{RES}$. Avant de créer une nouvelle étiquette L_v, on doit s'assurer que $r^{\text{dem}}(L_u) + \tau_{uv}^{\text{dem}}$, la demande cumulée espérée le long du chemin jusqu'au nœud v soit dans l'intervalle $[a_v^{\text{dem}}, b_v^{\text{dem}}] = [\mu_v, \mu_v]$ où $\mu_v \in \{E[\xi_j], ..., Q - E[\xi_j]\}$. On doit également s'assurer que $r^{\text{visit}_j}(L_u) = 0$ pour le client j. On continue jusqu'à ce que $a_w^{\text{dem}} \geq \frac{Q}{2}$ pour un nœud $w \in \mathcal{NS}$ donné ; c.-à.-d. jusqu'à ce que l'on soit certain que l'on ait consommé au moins la moitié de la capacité pour tout chemin de os à w.

On procède ensuite de manière similaire pour l'algorithme « arrière », mais en partant du puits os'. On fixe une étiquette $L_{os'}$ ayant une consommation de ressource nulle pour toutes les ressources. On fait progresser cette étiquette vers l'avant en augmentant la demande cumulée à chaque fois que l'on visite un nouveau prédécesseur. Comme pour l'algorithme avant, on doit s'assurer que les fenêtres de ressources sont respectées, c.-à.-d. $r^{\text{dem}}(L_v) + \tau_{vu}^{\text{dem}} \leq Q$ si on étend l'étiquette L_u basée au nœud $u = \mathscr{V}(\mu_i, i) \in \mathcal{NS}$, si on emprunte l'arc (v, u) vers l'arrière. On doit également s'assurer que $r^{\text{visit}_j}(L_u) = 0$ pour le noeud $v = \mathscr{V}(\mu_j, j)$. On ne prolonge pas une étiquette si $b_w^{\text{dem}} < \frac{Q}{2}$ pour un nœud w donné ; c.-à.-d. jusqu'à ce que l'on soit certain qu'il ne reste pas plus de la moitié de la capacité que l'on puisse consommer pour tout chemin arrière de os' à w. De manière équivalente, ceci signifie que $Q - b_w^{\text{dem}} > \frac{Q}{2}$. En d'autres mots, la borne inférieure sur la « capacité résiduelle » disponible pour tout chemin avant de os à w est strictement supérieure à $\frac{Q}{2}$. $a_w^s < \frac{Q}{2}$, car $a_w^s \leq b_w^s \ \forall s \in \mathcal{RES}, w \in \mathcal{NS}$.

Comme on arrête l'algorithme avant lorsque $a_w^s \geq \frac{Q}{2}$, ceci veut dire que l'on est certain que les chemins avant et arrière se rejoindront à un certain nœud et qu'il sera possible de les combiner pour créer des chemins de os à os'.

On observe que contrairement au cas « avant », si l'étiquette L_u a comme noeud terminal $\mathscr{V} = (\mu_i, i)$, $r^{\text{dem}}(L_u)$ ne correspond pas nécessairement à μ_i ni à $Q - \mu_i$. Il est donc nécessaire d'utiliser un ressource pour représenter la demande cumulée explicitement, car le graphe espace-état ne contient pas toute l'information nécessaire.

On fusionne ensuite les chemins partiels si ces derniers sont réalisables et on applique une règle de dominance modifiée afin de conserver uniquement les étiquettes Pareto-optimales correspondant à des chemins de la source au puits avec une consommation donnée de ressources. L'algorithme bidirectionnel permet ainsi de limiter le nombre d'étiquettes générées tout en garantissant l'obtention d'un chemin de la source au puits optimal.

4.2.6 Modification de l'ordre d'extension

Tel qu'expliqué dans (Irnich et Desaulniers, 2005) et (Feillet *et al.*, 2004), l'ordre de traitement des nœuds a une importance significative sur le temps total de traitement. Cet ordre est grandement influencé par la structure du graphe sous-jacent ainsi que les propriétés des fonctions d'extension.

Les algorithmes de (Desrochers *et al.*, 1992), (Feillet *et al.*, 2004) et (Righini et Salani, 2006) peuvent visiter un nœud donné plusieurs fois. Afin de gérer ces cycles potentiels, leurs algorithmes utilisent une liste de nœuds actifs contenant initialement la source. La procédure ajoute ensuite de nouveaux nœuds à cette liste lorsqu'elle prolonge de nouvelles étiquettes utiles vers ces nœuds. L'algorithme se termine lorsque cette liste est vide.

L'ordre de traitement des nœuds est donc déterminé par l'ordre de retrait de nœuds de cette liste. Par exemple, (Desrochers *et al.*, 1992) choisissent les nœuds de la liste associés aux étiquettes de valeur lexicographique minimale. Cet ordre, initialement utilisé dans (Desrochers et Soumis, 1988), peut être énoncé comme suit : soit $\mathbf{r}(L_1) = (r^{res_1}(L_1), r^{res_2}(L_1), ..., r^{res_R}(L_1))$ et $\mathbf{r}(L_2) = (r^{res_1}(L_2), r^{res_2}(L_2), ..., r^{res_R}(L_2))$, deux vecteurs de ressources associés à des étiquettes L_1 et L_2 différentes mais ayant le même nœud terminal où $\{res_1, res_2, ..., res_R\}$ représente l'ensemble des ressources considérées, alors l'étiquette L_1 vient avant L_2 dans l'ordre lexicographique (dénoté $\mathbf{r}(L_1) \prec_{lex} \mathbf{r}(L_2)$) si $\exists\ m > 0\ :\ r^{res_i}(L_1) = r^{res_i}(L_2)\ \forall i < m \wedge r^{res_m}(L_1) < r^{res_m}(L_2)$. Intuitivement, cet ordre indique que l'on choisit les étiquettes

ayant les plus petites consommation de ressources avant. Ceci se révèle essentiel pour la performance de leur algorithme. En utilisant une structure de donnée basée sur les « buckets » (un tableau dont l'indice de chaque élément indique l'inclusion dans un certain intervalle), ces auteurs sont capables de traiter chaque intervalle (élément du tableau correspondant à une liste chaînée d'étiquettes dont la consommation est dans un certain intervalle) une et une seule fois.

Il est possible d'utiliser un ordre différent lorsque le graphe est acyclique. Dans ce cas, il est effectivement possible de trier les nœuds selon l'ordre topologique et de faire progresser les étiquettes « avant » aux différents nœuds en considérant cet ordre. On peut montrer par induction qu'à chaque fois que l'algorithme termine de traiter un nœud donné avec cet ordre, tous les chemins partiels de la source à ce nœud ont été considérés.

On utilise l'ordre inverse du tri topologique pour considérer les nœuds lors de l'algorithme arrière. Il est également possible de montrer que dans ce cas, l'algorithme visitera un nœud u seulement si tous ses successeurs ont déjà été visités. De façon similaire au cas « avant », après avoir tiré toutes les étiquettes des successeurs de u vers ce nœud, on est assuré d'avoir généré tous les chemins partiels du puits à u.

On note que les nouvelles étiquettes générées à un nœud donné après avoir étendu les étiquettes de tous ses prédécesseurs (successeurs) seront permanentes pour la méthode avant (arrière). Voilà pourquoi cet algorithme est un algorithme de fixation d'étiquettes plutôt qu'un algorithme de modification itérative des étiquettes comme le sont les algorithmes de Bellman-Ford pour le SPP et ceux de (Feillet *et al.*, 2004) et (Righini et Salani, 2006) pour le SPPRC.

4.2.7 Modification pour prendre en compte l'élimination des 2-cycles

Comme $SP(non-elem)$, le sous-problème initial défini par les relations (3.43) à (3.46) n'impose pas de restrictions sur la présence de cycles, une approche naïve consiste à résoudre un SPPRC pouvant contenir des cycles de grandeur arbitraire.

Il est possible de renforcer cette relaxation en éliminant les 2-cycles (cycles du type (i, j, i) dans $\mathcal{G}$ où $i, j \in \mathcal{N}$). Ce type de plus court chemin a été introduit dès les années 1980 par (Houck *et al.*, 1980) et est notamment utilisé par (Christiansen et Lysgaard, 2007). Cette modification est relativement facile à implémenter. En effet, *si on n'empêche pas explicitement le retour au prédécesseur en fixant la ressource de visite à 1*, il est suffisant de conserver la

meilleure (ayant le plus petit coût réduit) et la seconde meilleure étiquette ayant un prédécesseur différent mais arrivant au même nœud donné. Toute étiquette associée à ce nœud final et ayant un coût supérieur au coût de ces deux étiquettes sera forcément dominée par l'une de ces étiquettes. En effet, on est certain que l'extension d'au moins l'une de ces étiquettes dominantes dominera l'extension de la nouvelle étiquette de coût supérieur. Il n'est donc plus nécessaire de conserver celle de coût supérieur.

Cette méthode n'augmente pas la complexité théorique des calculs nécessaires à la résolution du sous-problème par rapport à un SPPRC standard (Irnich et Villeneuve, 2004). L'élimination de 2-cycles nécessite effectivement au plus deux fois plus d'étiquettes qu'un algorithme de résolution de SPPRC standard à chaque nœud pour garantir une solution optimale. Ce sous-problème peut donc également être résolu en temps pseudo-polynomial et il est préférable de résoudre le sous-problème en considérant l'ensemble des routes réalisables ne contenant pas de 2-cycles.

4.2.8 Modification pour prendre en compte les voisinages (ng-routes)

Il est possible de renforcer SPPRC-2-cyc en éliminant les cycles de taille 3 ou plus. Si on élimine les cycles de taille $\kappa \geq 3$, on obtient ainsi un SPPRC-κ-cyc. Toutefois, ceci nécessite l'utilisation de structures de données complexes et augmente considérablement la complexité selon la taille des cycles éliminés (Irnich et Villeneuve, 2004).

Afin de contourner ce problème, il est possible d'utiliser un sous-problème produisant des routes se rapprochant des chemins élémentaires en facilitant l'élimination de petits cycles ayant de grandes chances de se retrouver dans la solution optimale de la relaxation linéaire du problème maître. On appellera ce sous-problème un SPPRC avec voisinage, car on interdit le retour à un ensemble de clients déterminé de façon dynamique selon les clients visités par le chemin ainsi que leur voisinage. Contrairement au SPPRC-κ cycle et au ESPPRC, il est impossible de garantir l'existence de cycles de grandeur fixe (à moins de combiner le SPPRC avec voisinage avec l'élimination de 2-cycles). Contrairement à la méthode de (Irnich et Villeneuve, 2004), cette méthode évite cependant de faire augmenter considérablement le temps de calcul par rapport à un SPPRC standard.

Ce nouveau sous-problème se base sur le concept de ng-routes proposé par (Baldacci *et al.*, 2011). Cette approche consiste d'abord à fixer un voisinage $\mathcal{N}_i$ de cardinalité $|\mathcal{N}_i| \leq \Delta(\mathcal{N}_i)$ pour chacun des clients i où $\Delta(\mathcal{N}_i)$ représente une borne supérieure fixe préétablie sur la taille du voisinage du client i. Pour notre implémentation, on fixe toujours la cardina-

lité du voisinage à cette borne, c.-à-d. $|(\mathcal{N}_i)| = \Delta(\mathcal{N}_i)\ \forall i \in \mathcal{N}$. Pour une route partielle $p = (o = i_1, i_2, ..., i_k)$ donnée, on interdit ensuite le retour aux clients qui appartiennent à l'ensemble $\Pi(p) = \{i_r : i_r \in \cap_{s=r+1}^{k} \mathcal{N}_{i_s}, r = 1, ..., k-1\} \cup \{i_k\}$ où $\mathcal{N}_i$ représente le voisinage du client i. Ainsi, $\Pi(p)$ représente l'ensemble des clients précédant i_k qui appartiennent au voisinage de tous leurs clients successeurs dans cette route. Il existe plusieurs façons de définir $\mathcal{N}_i$, le voisinage du nœud i. Dans notre implémentation, nous adoptons la construction suivante :

Soit :

- $\mathcal{GS}^l = (\mathcal{NS}^l, \mathcal{AS}^l)$: le graphe résiduel au l^e nœud de branchement. où $\mathcal{NS}^l \subseteq \mathcal{NS}$ et $\mathcal{AS}^l \subseteq \mathcal{AS}$.
- $\zeta_{ij}^l = c_{ij} + \text{EFC}(\underline{\mu}_j, j)$ où $\underline{\mu}_j = \min_{\mu_i \in \tilde{\mu}(i)^l}\{\mu_i + E[\xi_j]\}$ et $\tilde{\mu}(i)^l$ représente la plus petite demande cumulée que l'on puisse atteindre au client j si l'on passe par le client i avant dans le graphe $\mathcal{GS}^l$.

ζ_{ij}^l représente donc le plus petit coût de déplacement espéré entre le client i et le client j si le client i peut être atteint à partir de i dans le l^e nœud de branchement. ζ_{ij}^l peut donc être interprété comme une mesure « d'attirance » de i vers j.

Si $\prec$ définit une relation d'ordre sur les éléments de S_i et que $j_k \prec j_{k+1}$ indique que j_k vient avant j_{k+1}, alors on obtient :

- $S_i = \{j_1, j_2, ..., j_{|\mathcal{N}|-1} \mid j_k \prec j_{k+1} \text{ si } \zeta_{ij_k}^l \not\succ \zeta_{ij_{k+1}}^l\} \subseteq \mathcal{N} \setminus \{i\}$: les $|\mathcal{N}|-1$ clients n'incluant pas i triés en ordre non décroissant de ζ_{ij}^l.
- $\mathcal{N}_i = \{j_1, ..., j_{\Delta(\mathcal{N}_i)-1} \mid j_k \in S_i\ \forall k\} \cup \{i\}$: les $\Delta(\mathcal{N}_i) - 1$ plus petits éléments de S_i ainsi que i lui-même.

Plusieurs travaux suggèrent qu'il est utile de considérer le voisinage comme les $\Delta(\mathcal{N}_i)$ clients les plus près. Des résultats empiriques préliminaires ont toutefois démontré qu'il était encore plus efficace de considérer les voisinages selon les clients les plus près en distance espérée que seulement selon la distance déterministe.

Étant donné $p = \{o = i_1, i_2, ..., i_r, ..., i_k\} \subseteq \mathcal{N}$, on ne pourra revisiter i_r si ce dernier appartient à tous les voisinages de ses successeurs. De manière équivalente, on accepte de créer un cycle en revisitant le client i_r seulement si ce dernier n'est pas dans au moins un des voisinages de ses $k - r$ successeurs. Pour être inclus dans $\Pi(p)$, un client visité plus tard doit

appartenir à un plus petit nombre de voisinages qu'un nœud visité plus tôt. Par conséquent, un nœud visité plus tard a de plus grandes chances d'appartenir à l'ensemble interdit $\Pi(p)$. Intuitivement, les ng-routes permettent donc de réduire le nombre de petits cycles dans les routes.

Il est important de noter que le borne produite par l'utilisation des ng-routes n'est pas nécessairement plus élevée que celle produite par l'élimination de 2 cycles. Considérons l'exemple suivant :

$\mathcal{N} = \{1, 2, 3, 4, 5\} \cup \{o\}$: l'ensemble des clients et le dépôt
$p = (o, 1, 2, 1)$: un chemin partiel
$\Delta(\mathcal{N}_i) = 3 \; \forall i \in \mathcal{N}$

$\mathcal{N}_1 = \{1, 5, 6\}$
$\mathcal{N}_2 = \{2, 5, 6\}$
$\mathcal{N}_3 = \{3, 4, 5\}$
$\mathcal{N}_4 = \{4, 2, 3\}$
$\mathcal{N}_5 = \{5, 2, 3\}$

$\Pi(p' = (o, 1, 2)) = \{2\}$ car $1 \notin \mathcal{N}_1 \cap \mathcal{N}_2$

p sera donc réalisable avec les ng-routes, mais pas avec l'élimination de 2-cycles. Nous illustrons cette relations dans les figures de la section 4.2.14. Dans la figure 4.5, nous constatons notamment que la valeur de la relaxation linéaire est plus élevée avec élimination de 2-cycles qu'avec uniquement l'utilisation de voisinages de taille 5 ; et ce pour la majorité des instances testées. Dans les sections de résultats, nous avons donc combiné les ng-routes à l'élimination de 2-cycles afin de garantir une solution au moins aussi bonne que celle produite par la résolution d'un SPPRC-2-cyc.

Bien que $\mathcal{N}_i$ puisse changer à chaque nœud de l'arbre de branchement à cause des décisions de branchement ou des inégalités valides ajoutées, on note que tout chemin non réalisable à un nœud de branchement le sera également dans tous les nœuds de branchement fils.

On note également qu'augmenter la valeur de $\Delta(\mathcal{N}_i) \; \forall i \in \mathcal{N}$ n'augmente pas forcément la valeur optimale du sous-problème. Toutefois, si on définit toujours les voisinages selon les mêmes critères et qu'on considère deux instances $\mathcal{I}_1$ et $\mathcal{I}_2$ du même problème de plus court

chemin avec $\mathcal{N}_i^{\mathcal{I}_1} \subseteq \mathcal{N}_i^{\mathcal{I}_2},\ \forall i \in \mathcal{N}$, alors $\mathcal{I}_1$ sera nécessairement une relaxation de l'instance $\mathcal{I}_2$ avec un plus grand voisinage. Cette relation est illustrée dans la figure 4.5 de la section 4.2.14 . On constate effectivement que les courbes correspondant aux ng-routes avec de plus grands voisinages sont toujours égale ou au-dessus des courbes avec plus petit voisinage.

Par ailleurs, on est assuré d'augmenter la valeur de la solution du sous-problème si on augmente suffisamment la valeur de $\Delta(\mathcal{N}_i)\ \forall i$ par rapport à $|\mathcal{N}| = n$. Par exemple, si on fixe $\Delta(\mathcal{N}_i) = |\mathcal{N}|\ \forall i$, alors on est certain d'obtenir une borne inférieure identique à celle obtenue par le sous-problème élémentaire. Dans ce cas, chaque client sera forcément dans le voisinage de tous les clients. Chaque chemin partiel ne pourra donc pas retourner aux clients qu'il a déjà visités, car ces derniers sont dans le voisinage de tous ses successeurs.

Des résultats empiriques (voir 4.2.14) pour cette formulation du VRPSD ainsi que les travaux de (Baldacci *et al.*, 2011) démontrent que fixer $\Delta(\mathcal{N}_i) = 10,\ \forall i \in \mathcal{N}$, permet de tirer un bon compromis entre la rapidité de résolution du sous-problème et la valeur de la borne inférieure produite à un nœud de l'arbre de branchement.

4.2.9 Modification pour prendre en compte les contraintes d'élémentarité

Il est également possible de renforcer le SPPRC avec élimination de 2-cycles et voisinage. Comme les contraintes (3.40) du problème maître impliquent que toute route dans la solution optimale de PM devra visiter chaque client au plus une fois. Il est effectivement possible de fixer $\Lambda = \mathscr{PE}$ où $\mathscr{PE}$ représente l'ensemble des routes de os à os' réalisables et élémentaires, c.-à-d. les routes qui ont une demande cumulée espérée inférieure ou égale à la capacité du véhicule et qui visitent un client au plus une fois. On peut donc renforcer le sous-problème original et utiliser un problème de plus court chemin avec contraintes de ressources élémentaire (« elementary shortest path problem with resource constraints » - ESPPRC). Ceci est équivalent à ajouter les $2^{\mathcal{N}'}$ contraintes d'élimination de sous-tours suivantes au sous-problème défini par 3.43 à 3.46 :

$$\sum_{\substack{(a,b)\in\mathcal{AS} \\ a\in C(i), b\in C(j) \\ i,j\in S}} \hat{x}_{ab} \leq |S| - 1 \quad \forall S \subseteq \mathcal{N}'$$

Par conséquent, on ne peut se limiter à résoudre un SPPRC standard sur $\mathcal{GS}$, car ce type de problème permettrait la génération de routes avec des cycles (visitant un client plus d'une fois). En effet, même si $\mathcal{GS}$ est acyclique, le graphe original $\mathcal{G}$ est cyclique et il est possible

de retourner à un client donné avec une demande cumulée différente dans $\mathcal{GS}$.

La résolution d'un ESPPRC, correspond à la résolution d'un SPPRC-κ-cyc avec $\kappa = |\mathcal{N}|$. On s'assure ainsi d'éliminer tous les cycles possibles. (Beasley et Christofides, 1989) ont notamment proposé une technique pour résoudre un tel problème en ajoutant $|\mathcal{N}|$ ressources binaires de la forme $r^{\text{visit}_j}(L_u)\ \forall j \in \mathcal{N}$ indiquant si un client j peut être visité par l'extension d'un chemin partiel associé à l'étiquette L_u basée au nœud $u = \mathscr{V}(\mu_i, i) \in C(i)$. On pose $\tau_{uv}^{\text{visit}_j} = 1\ \forall (u,v) \in \mathcal{AS}$ et $\tau_{uw}^{\text{visit}_h} = 0\ \forall (u,w) \in \mathcal{AS}$ où $v = \mathscr{V}(\mu_j, j) \in C(j)$, $w = \mathscr{V}(\mu_h, h) \in C(h); h \in \mathcal{N}; h \neq j \neq i$ et $a_u^{visit_j} = 0, b_u^{visit_j} = 1\ \forall j \in \mathcal{N}, u \in \mathcal{N}$. On « active » donc la ressource de visite associée à un client j si et seulement si on visite un nœud dans l'ensemble $C(j)$.

Bien que les travaux de (Desrochers et Soumis, 1988) démontrent qu'il est possible de résoudre le SPPRC à l'aide d'un algorithme pseudo-polynomial, (Dror, 1994) prouve que le problème de plus court chemin avec contraintes de ressources élémentaire (ESPPRC) est fortement $\mathcal{NP}$-difficile et qu'il n'existe pas d'algorithme polynomial pour le résoudre (à moins que $\mathcal{P} = \mathcal{NP}$). On note notamment que le principe de Bellman tel qu'énoncé pour le SPPRC ne s'applique plus, car il ne suffit pas de résoudre un SPPRC sans ressources de visite et de conserver uniquement les chemins élémentaires pour garantir l'optimalité de la solution. Si on résout le ESPPRC en ajoutant n ressources de visite binaires, on doit alors considérer l'ensemble des vecteurs de ressource Pareto optimaux selon cette nouvelle définition pour chaque nœud à chaque étape. Dans ce cas, on obtient néanmoins $O(Q2^n)$ états à chaque nœud et le nombre d'étiquettes augmente très rapidement. Ce type de sous-problème est donc beaucoup plus difficile à résoudre. La section suivante décrit avec plus de détails l'algorithme utilisé dans notre implémentation.

4.2.10 Description de l'algorithme final

Avant de décrire plus en détail l'algorithme (E)SPPRC bidirectionnel, il est nécessaire de rappeler ou d'introduire la notation suivante :

Soit :
- $\mathcal{RES}$ où $|\mathcal{RES}| = R$: l'ensemble des R ressources considérées par le problème ;
- $L_u = (\bar{c}, \mathbf{r}, \bar{v})$: une étiquette représentant un chemin partiel de la source ou du puits au nœud $u \in \mathcal{VS}$ et auquel est associé un chemin unique dans $\mathcal{G}$;
- $\bar{v}(L_u) = u$: le nœud final dans $\mathcal{GS}$ du chemin partiel associé à l'étiquette L_u ;

- $pred(L_u)$: l'avant-dernier *nœud* du graphe $\mathcal{GS}$ visité par le chemin partiel associé à l'étiquette L_u (s'il existe) ;
- $Client(u) = i$: le client $i \in \mathcal{N}$ associé au noeud $u \in \mathcal{NS}$;
- $c(L_u)$: le coût espéré total du chemin partiel ;
- $\bar{c}(L_u)$: le coût réduit espéré total du chemin partiel ;
- $\mathbf{r}(L_u) = (r^s(L_u)) \in \mathbb{R}^R$: le vecteur indiquant le niveau de consommation des différentes ressources pour l'étiquette L_u ;
- τ^s_{uv} : une borne inférieure sur la consommation de la ressource s lorsque l'arc $(u, v) \in \mathcal{AS}$ est emprunté ;
- $\Pi(L)$: par abus de langage, les clients dont on interdit la visite pour le chemin partiel associé à l'étiquette L à cause des ng-routes ;
- $\epsilon^{fw}_u = (Client(u), ..., o')$ une extension réalisable vers l'avant du chemin associé à l'étiquette L_u. Une extension représente donc un chemin partiel de $Client(u)$ à o' **dans** $\mathcal{G}$;
- $\epsilon^{bw}_u = (Client(u), ..., o)$ une extension réalisable vers l'arrière du chemin associé à l'étiquette L_u ;
- $L \oplus_{\mathcal{G}} \epsilon = L'$: par abus de langage, l'opérateur $\oplus_{\mathcal{G}}$ permet d'obtenir *l'étiquette* associée au *chemin* partiel obtenu par l'extension du chemin associé à l'étiquette L à l'aide de l'extension ϵ dans $\mathcal{G}$;
- $L \oplus_{\mathcal{GS}} p = L'$: l'opérateur $\oplus_{\mathcal{GS}}$ permet d'obtenir *l'étiquette* associée au *chemin* partiel obtenu par l'extension du chemin associé à l'étiquette L à l'aide du chemin p dans $\mathcal{GS}$;
- $\mathcal{E}^{fw}(L_u)$: l'ensemble des extensions réalisables vers l'avant pour le chemin associé à l'étiquette L_u ;
- $\mathcal{E}^{bw}(L_u)$: l'ensemble des extensions réalisables vers l'arrière pour le chemin associé à l'étiquette L_u ;
- $\mathcal{F}_{\mathcal{GS}}(u, v)$: l'ensemble des chemins associés à des étiquettes représentant un chemin de u à v dans $\mathcal{GS}$ réalisable par rapport à la consommation des ressources ;
- $\mathcal{S}_{\mathcal{GS}}$: l'ensemble des chemins dans $\mathcal{GS}$ associés à des étiquettes représentant un chemin réalisable par rapport à l'élimination des cycles ;
- $\mathcal{T}_{\mathcal{GS}}(p) = \{\mathbf{r}_{|\mathbf{p}|} : \exists \mathbf{r_i} \in [a_i, b_i] \mathbf{f_{v_i v_{i+1}}}(\mathbf{r_i}) \leq \mathbf{r_{i+1}} \ \forall i \in \{1, 2, ..., |p| - 1\}\}$: l'ensemble des vecteurs de ressource réalisables pour le chemin $p = \{v_1, v_1, ..., v_{|p|}\} \subseteq \mathcal{NS}$;
- Γ^{fw}_u : l'ensemble des étiquettes avant « utiles » à un nœud $u \in C(i)$, c.-à.-d. les chemins de os à u pouvant mener à des chemins de os à os' optimaux ;
- Γ^{bw}_u : l'ensemble des étiquettes arrières « utiles » à un nœud $u \in C(i)$, c.-à.-d. les chemins de os' à u pouvant mener à des chemins de os' à os optimaux ;
- $Topo^{fw}$: une liste de nœuds triés selon l'ordre topologique du graphe $\mathcal{GS}$;

- $Topo^{bw}$: une liste de nœuds triés selon l'ordre inverse de l'ordre topologique du graphe $\mathcal{GS}$;
- $ArcsTemporaires$: une liste temporaire d'arcs utilisés afin de réaliser le tri topologique ;
- SP_{type} : une variable qui indique le type de sous-problème utilisé ;
- $f^s_{uv}(\mathbf{r}(L_u)) : \mathbb{R}^R \to \mathbb{R}$: la fonction d'extension qui indique une borne inférieure sur la consommation totale de la ressource $s \in \mathcal{RES}$ le long de l'arc $(u = (\mu_i, i), v = (\mu_j, j)) \in \mathcal{AS}$ si on a consommé $\mathbf{r}(L_u)$ le long du chemin partiel correspondant à l'étiquette L_u.

Algorithme 3: (E)SPPRC($\mathcal{GS}, S^l_{IP,k}, SP_{type}, \pi_k$)

ModifierGraph($\mathcal{GS}, SP_{type}, \pi_k$)
$\Gamma^{fw}_{os} = (0, \mathbf{0}, os)$, $\Gamma^{bw}_{os'} = (0, \mathbf{0}, os')$
pour chaque $u \in \mathcal{NS} \setminus \{os\}$ **faire** $\Gamma^{fw}_u \leftarrow \emptyset$
pour chaque $u \in \mathcal{NS} \setminus \{os'\}$ **faire** $\Gamma^{bw}_u \leftarrow \emptyset$
$ArcsTemporaires \leftarrow \emptyset$
pour chaque $(\mathscr{V}(i,\mu_i), \mathscr{V}(j,\mu_j)) \in \mathcal{AS} \mid i,j \notin \{o,o'\}$ **faire**
  si $i = j$ **et** $\mu_i < \mu_j$ **alors**
    $\mathcal{AS} \leftarrow \mathcal{AS} \cup (\mathscr{V}(i,\mu_i), \mathscr{V}(j,\mu_j))$
    $ArcsTemporaires \leftarrow ArcsTemporaires \cup (\mathscr{V}(i,\mu_i), \mathscr{V}(j,\mu_j))$
  fin
fin
$Topo^{fw} \leftarrow$**TriTopologique**($\mathcal{GS}, \mathcal{AS}$)
$Topo^{bw} \leftarrow$**Inverse**($Topo^{fw}$)
pour chaque $(\mathscr{V}(i,\mu_i), \mathscr{V}(j,\mu_j)) \in TempArc$ **faire**
  $\mathcal{AS} \leftarrow \mathcal{AS} \setminus (\mathscr{V}(i,\mu_i), \mathscr{V}(j,\mu_j))$
fin
pour chaque $v = \mathscr{V}(j,\mu_j) \in Topo^{fw}$ **faire**
  si $a^{dem}_v \geq \frac{Q}{2}$ **ou** $(\delta^+(\{v\}) = \emptyset$ **et** $v \neq os')$ **alors continue**
  pour chaque $u = \mathscr{V}(i,\mu_i) : (u,v) \in \delta^-(\{v\})$ **faire**
    pour chaque $L_u \in \Gamma^{fw}_u$ **faire**
      si $(SP_{type} = elem$ **et** $r^{visit_j}(L_u) = 1)$
        ou $(SP_{type} = 2\text{-}cyc$ **et** $pred(L_u) \in C(j))$
        ou $(SP_{type} = ng\text{-}route$ **et** $j \in \Pi(L_u)$ **et** $r^{visit_j}(L_u) = 1)$ **alors**
      continue
      sinon
        $L_v \leftarrow$ **ÉtendreAvant**(L_u, v)
        $\Gamma^{fw}_v \leftarrow$ **InsérerDominerAvant**(Γ^{fw}_v, L_v)
      fin
    fin
  fin
fin
pour chaque $v = \mathscr{V}(j,\mu_j) \in Topo^{bw}$ **faire**
  si $b^{dem}_v < \frac{Q}{2}$ **ou** $(\delta^-(\{v\}) = \emptyset$ **et** $v \neq os)$ **alors continue**
  pour chaque $u = \mathscr{V}(i,\mu_i) : (v,u) \in \delta^+(\{v\})$ **faire**
    pour chaque $L_u \in \Gamma^{bw}_u$ **faire**
      si $(SP_{type} = elem$ **et** $r^{visit_j}(L_u) = 1)$
        ou $(SP_{type} = 2\text{-}cyc$ **et** $pred(L_u) \in C(j))$
        ou $(SP_{type} = ng\text{-}route$ **et** $j \in \Pi(L_u)$ **et** $r^{visit_j}(L_u) = 1)$ **alors**
      continue
      sinon
        $L_v \leftarrow$ **ÉtendreArrière**(L_u, v)
        $\Gamma^{bw}_v \leftarrow$ **InsérerDominerArrière**(Γ^{bw}_v, L_v)
      fin
    fin
  fin
fin
retourne Joindre(Γ^{fw}, Γ^{bw})

La méthode **ModifierGraph**$(\mathcal{GS}, SP_{type}, \pi_k)$ modifie le graphe $\mathcal{GS}$ en fonction des variables duales trouvées à cette itération de génération de colonnes ainsi que les inégalités valides ajoutées et les décisions de branchement.

Comme on le note aux lignes 17 et 31, on ne considère pas un nœud s'il n'est pas le puits os' et n'a pas de nœuds successeurs (si on utilise l'algorithme avant) et qu'on ne considère pas un nœud si ce dernier n'est pas la source os et n'a pas de prédécesseur (si on utilise l'algorithme arrière). En effet, il est inutile de tirer des étiquettes vers un nœud inutile ne pouvant se retrouver dans un chemin de la source au puits.

De plus, ces lignes s'assurent que l'on considère les nœuds dans l'ordre topologique « avant » (« arrière ») et on tire les étiquettes des nœuds prédécesseurs (successeurs) vers les nœuds considérés à l'itération donné. Les lignes 18 et 32 s'assurent que l'on arrête les algorithmes avant et arrière à mi-chemin tout en s'assurant que l'on pourra toujours obtenir des chemins complets de la source au puits après avoir fusionné les deux segments grâce à la méthode **Joindre** $(\Gamma^{fw}, \Gamma^{bw})$. Cette procédure fusionne les étiquettes avant et arrière associées au même nœud final. Cette procédure s'assure ensuite que les nouveaux chemins de os à os' ainsi créés soient réalisables. Elle filtre ensuite ces chemins pour considérer uniquement ceux de coût réduit négatif. Il est donc possible d'ajouter plusieurs colonnes de coût réduit négatif au problème maître à la fin de la résolution de chaque (E)SPPRC.

La méthode **ÉtendreAvant** (L_u, v) étend l'étiquette L_u vers l'avant le long de l'arc $(u, v) \in \mathcal{AS}$ alors que **ÉtendreArrière** (L_u, v) l'étend vers l'arrière le long de l'arc (v, u). Dans les deux cas, l'extension est un processus qui implique la prolongation du chemin partiel se terminant actuellement au nœud u à l'aide d'un nouvel arc $(u, v) = (\mathscr{V}(\mu_i, i), \mathscr{V}(\mu_j, j)) \in \mathcal{AS}$; où $i, j \in \mathcal{N}$; $\mathscr{V}(\mu_i, i), \mathscr{V}(\mu_j, j) \in \mathcal{NS}$. Cette procédure met également à jour le nœud final du nouveau chemin partiel, le coût réduit du chemin, le nœud final du chemin et le vecteur de consommation de ressources.

Comme l'extension des étiquettes vers l'avant et vers l'arrière est très semblable, on omet les exposant fw et bw dans la notation qui suit afin d'alléger la lecture. On note par ailleurs que les consommations de ressources sont indépendantes entre elles. On écrit donc la fonction d'extension $f^s_{uv}(r^s(L_u))$ comme une fonction de la ressource s particulière, plutôt que du vecteur de consommation entier - $f^s_{uv}(\mathbf{r}(L_u))$.

Si on étend l'étiquette L_u vers v, alors on créera la nouvelle étiquette L_v et on aura

(bien que l'on ne considère pas $\bar{v}$, le nœud final comme une ressource, on écrit par abus de notation) :

$$f_{uv}^{\bar{v}}(\bar{v}(L_u)) = v \tag{4.3}$$

Le nouveau coût réduit et la consommation des autres ressources sont quant à eux déterminés à l'aide des fonctions d'extension suivantes :

$$f_{uv}^{\bar{c}}(\bar{c}(L_u)) = \bar{c}(L_u) + \bar{c}_{uv} \tag{4.4}$$

$$f_{uv}^{\text{dem}}(r^{\text{dem}}(L_u)) = r^{\text{dem}}(L_u) + E[\xi_j] \tag{4.5}$$

où on suppose qu'on a accumulé $\mu_i = r^{\text{dem}}(L_u)$ au nœud $u \in C(i)$ et $\mu_j = \mu_i + E[\xi_j]$ au nœud $v \in C(j)$.

Si on utilise des ressources binaires pour indiquer si on a visité un client lors de la résolution du (E)SPPRC, alors on aura également :

$$f_{uv}^{\text{visit}_h}(r^{\text{visit}_h}(L_u)) = \begin{cases} r^{\text{visit}_h}(L_u) + 1 & \text{si } h = j \\ r^{\text{visit}_h}(L_u) & \text{sinon} \end{cases} \tag{4.6}$$

où $h \in \mathcal{N}, h \neq i \neq j$.

Si on utilise un SPPRC avec voisinage, alors on fixera une ressource de visite à 1 seulement si le client associé fait partie du chemin partiel associé à L_u et s'il fait également partie des clients interdits - $\Pi(L_u)$.

$$r^{\text{visit}_h}(L_u) = \begin{cases} 1 & \text{si } h \in \Pi(L_u) \\ 0 & \text{sinon} \end{cases} \tag{4.7}$$

Ainsi, contrairement au ESPPRC, il est possible qu'une ressource de visite reprenne la valeur 0 si le client associé ne fait plus partie des noeuds interdits.

Afin de faciliter la dominance, on met également à jour les ressources de visite de façon

prévisionnelle. Cette méthode, proposée par (Feillet *et al.*, 2004), consiste à identifier les clients qui ne pourront pas être visités par le chemin partiel à cause de limitation sur la consommation totale de ressources. Il est notamment possible qu'un client ne puisse être visité à cause de contraintes de capacité. Par exemple, si une étiquette L_u associée à un chemin partiel de *os* à u n'a pas pas encore visité le client $h \in \mathcal{N} : h \neq i \neq j$, (c.-à.-d. $r^{\text{visit}_h}(L_u) = 0$), mais qu'il est impossible de visiter ce client à partir de cette étiquette, car une telle extension produirait un chemin dont la demande totale espérée est strictement supérieure à la capacité d'un véhicule, alors on peut quand même fixer cette ressource de visite à 1.

$$r^{\text{visit}_h}(L_u) = 1 \text{ si } r^{\text{dem}}(L_u) + E[\xi_h] > Q \tag{4.8}$$

Ceci permet de raffiner la définition du vecteur de ressources de visite. On considère effectivement les clients non atteignables comme ceux qui ne peuvent pas être visités par le chemin partiel plutôt que ceux qui sont déjà dans ce chemin. Cette modification permet d'éliminer plus de chemins inutiles.

(Feillet *et al.*, 2004) proposent d'effectuer cette mise à jour à chaque fois qu'une étiquette est étendue vers ses nœuds successeurs. Ils vérifient s'il existe une ressource $s \in \mathcal{RES}$ et un arc $(u, v) \in \mathcal{AS}$ sortant de $u = \bar{v}(L_u)$ tels que l'inégalité suivante est respectée $r^s(L_u) + \tau^s_{uv} > b^s_j$. Lorsqu'il existe beaucoup de ressources et que le temps de mise à jour devient trop important, ils proposent de considérer uniquement un sous-ensemble de ressources jugées importantes et permettant d'identifier des clients non atteignables (e.g. le temps cumulé et la demande cumulée). Pour notre problème, la demande cumulée est la seule ressource candidate, car on ne peut considérer le coût ni les ressources de visite. Il est donc facile d'appliquer cette procédure sans augmenter considérablement le temps de calcul de l'algorithme.

Cette règle peut être appliquée sous cette forme, car la consommation de la ressource de demande cumulée est non négative sur tous les arcs et respecte l'inégalité du triangle. Supposons qu'il est impossible de visiter le client h à partir de l'étiquette L_u à cause de la contrainte de capacité en utilisant uniquement le chemin $p^1_u = \{u, v\}$. Il est alors également impossible de le visiter à partir de tout chemin $p^2_u = \{u, \varrho_1, \varrho_2, ..., \varrho_{|p^2_u|-2}, w\}$ dans $\mathcal{GS}$ tel que $\varrho_\iota \in \mathcal{NS}, w \in \mathcal{NS}\ \forall \iota \in \{1, ..., |p^2_u| - 2\}$, $|p^2_u| > |p^1_u|$ et $w, v \in C(h)$. Comme on a $E[\xi_t] > 0\ \forall t \in \mathcal{N}$, on sait que $\sum_{\iota=1}^{|p^2_u|-2} E[\xi_{t_\iota}] > 0$ où $\varrho_\iota \in C(t_\iota)$. De plus, $f^{\text{dem}}_{\theta\vartheta}(r^{\text{dem}}(L_\theta)) = r^{\text{dem}}(L_\theta) + E[\xi_h]$ pour tout arc $(\theta, \vartheta) \in \mathcal{AS} : \theta \in C(g);\ g \in \mathcal{NS}, g \neq$

$h;\vartheta \in C(h)$. On est donc certain que tout arc ayant un nœud d'arrivée dans l'ensemble $C(h)$ consommera $E[\xi_h]$. Les arcs (u,v) et $(\varrho_{p_u^2}, w)$ ont donc tous deux une consommation de demande cumulée de $E[\xi_h]$ et les arcs $(\varrho_\iota, \varrho_{\iota+1}) : 1 \leq \iota \leq |p_u^2| - 3$ ont tous une consommation de demande cumulée non nulle. Tel qu'illustré dans la figure 4.1, on a donc forcément $r^{\text{dem}}(L_u \oplus_{\mathcal{GS}} p_u^1) < r^{\text{dem}}(L_u \oplus_{\mathcal{GS}} p_u^2)$. Comme (u,v) est la plus petite extension possible du nœud u à v, si $\exists (u,v) \in \mathcal{AS} : r^{\text{dem}}(L_u \oplus_{\mathcal{GS}} (u,v)) > Q$, alors $r^{\text{dem}}(L_u \oplus_{\mathcal{GS}} p_u) > Q$ pour tout chemin p_u se terminant au client h et le client h ne peut être atteint à partir d'aucun chemin à partir de l'étiquette L_u.

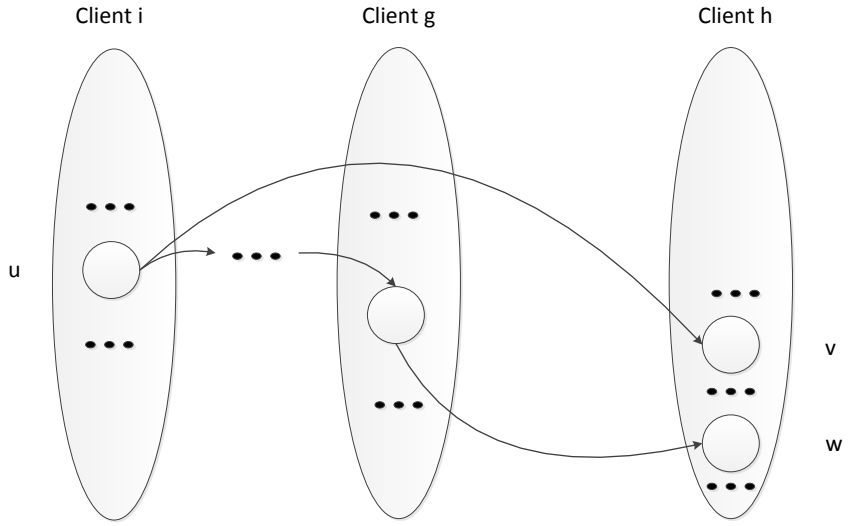

Figure 4.1 Respect de l'inégalité du triangle par la demande cumulée

Toutefois, les nœuds $u = \mathscr{V}(\mu_u, u)$ et $v = \mathscr{V}(\mu_j, j)$ dans le graphe $\mathcal{GS}$ représentent un état partiel *réalisable* pour un chemin donné au client i et j. En effet, on a $\mu_i \leq Q \; \forall i \in \mathcal{N}$, car on considère uniquement les chemins réalisables en moyenne. Ainsi, $r^{\text{dem}}(L_u) + \tau_{uv}^{\text{dem}} = \mu_i + E[\xi_j] \leq Q \; \forall (u,v) \in \mathcal{AS}$. Il devient donc inutile de vérifier le respect de cette inégalité.

Cependant, il est vrai que certaines étiquettes ne peuvent visiter des clients lorsque leur demande cumulée ou la demande espérée du nouveau client sont trop grandes par rapport à

la capacité résiduelle. Pour déterminer les clients non atteignables, il suffit de vérifier l'existence de l'arc $(\mathscr{V}(\mu_i, i), \mathscr{V}(\mu_i + E[\xi_h], h))$. Comme les règles de branchement peuvent avoir éliminé ou imposé certains arcs du graphe $\mathcal{GS}$ original, il est important de considérer $\bar{\mathcal{AS}}$; l'ensemble des arcs originaux de $\mathcal{GS}$.

Comme nous considérons également l'élimination de 2-cycles, on fixe la ressource de visite correspondant au prédécesseur de l'étiquette considérée à 1 ; $r^{\text{visit}_h}(L_u) = 1$ où $Client(pred(L_u)) = h$, et ce indépendamment du fait que cette ressource soit dans l'ensemble des clients interdits par les ng-routes ou non. Cette ressource de visite est fixée de manière dynamique à mesure que l'on prolonge les étiquettes. Il est important de réévaluer cette valeur aux prolongations suivantes, car on empêche uniquement le retour à un prédécesseur immédiat. Ceci permet de raffiner la fonction d'extension initiale pour la consommation de ressources de visite :

$$r^{\text{visit}_h}(L_u) = \begin{cases} 1 & \text{si } h = Client(pred(L_u)) \text{ ou } h \in \Pi(L_u) \\ 0 & \text{sinon} \end{cases} \tag{4.9}$$

La méthode **InsérerDominerX** (Γ_v, L_v) ajoute l'étiquette L_v à l'ensemble Γ_v si et seulement si cette dernière peut mener à un plus court chemin de la source au puits réalisable par rapport aux contraintes de consommation de ressources et d'élimination de cycles. Comme les étiquettes correspondent à des vecteurs de ressources, ceci revient à identifier des chemins Pareto optimaux de la source au puits.

On cherche à conserver le moins d'étiquettes possible afin d'accélérer la résolution de l'algorithme. Il est donc bénéfique d'éliminer les étiquettes qui ne peuvent mener à des chemins Pareto optimaux de la source au puits. On définit ces chemins comme suit :

Definition 1. *Chemins inutiles de u à v dans $\mathcal{GS}$ (Irnich et Villeneuve, 2004)*

Soit $L_v^m \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S},\ m = 1, ..., t$; des étiquettes associées à des routes réalisables par rapport aux cycles et la consommation de ressources. Considérons l'étiquette $L_v^0 \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S}_{\mathcal{GS}}$ qui répond aux critères suivants :

$$\bar{v}(L_v^0) = \bar{v}(L_v^m)\ \forall m \in 1, ...t \tag{4.10}$$

$$\bar{c}(L_v^m) \leq \bar{c}(L_v^0)\ \forall m \in 1, ...t \tag{4.11}$$

$$\mathcal{E}(L_v^0) \subseteq \cup_{m=1}^{t} \mathcal{E}(L_v^m) \tag{4.12}$$

Alors pour toute extension réalisable $\epsilon_0 \in \mathcal{E}(L_v^0)$ *de* L_v^0 *(c.-à.-d.* $L_v^0 \oplus_{\mathcal{G}} \epsilon_0 \in \mathcal{F}_{\mathcal{GS}}(v, os') \cap \mathcal{S}_{\mathcal{GS}}$*), on est capable de trouver au moins une étiquette* $L_v^{m^*}$ *où* $m^* \in \{1, ..., t\}$ *telle que :*

$$\bar{c}(L_v^{m^*} \oplus_{\mathcal{G}} \epsilon_0) \leq \bar{c}(L_v^0 \oplus_{\mathcal{G}} \epsilon_0) \tag{4.13}$$

$$L_v^{m^*} \oplus_{\mathcal{G}} \epsilon_0 \in \mathcal{F}_{\mathcal{GS}}(v, os') \cap \mathcal{S}_{\mathcal{GS}} \tag{4.14}$$

L'étiquette L_v^0 n'est donc pas nécessaire et on peut l'éliminer. Afin d'appliquer une telle règle d'élimination, on doit s'assurer que la fonction totale de coût d'un chemin (incluant le coût espéré d'échec) est croissant selon la demande cumulée le long de ce chemin. Nous prouvons en annexe que le coût espéré d'échec à un client est bien une fonction croissant de façon monotone selon la demande cumulée à ce client.

On remarque que l'identification de chemins inutiles pouvant être éliminés nécessite l'évaluation de $\mathcal{E}(L)$: l'ensemble des extensions pouvant découler d'un chemin associé à une étiquette L donnée. Il n'est pas nécessaire de codifier $\mathcal{E}(L)$ explicitement lorsqu'on cherche à éliminer des 2-cycles. En effet, il est suffisant de considérer deux étiquettes ayant le même noeud terminal et ayant des prédécesseurs différents pour couvrir l'ensemble des extensions possibles pour toutes les étiquettes ayant ce noeud terminal.

Cependant, lorsqu'on cherche à résoudre un SPPRC avec voisinage ou un ESPPRC, il est nécessaire de codifier $\mathcal{E}(L)$. Nous choisissons de représenter les extensions possibles d'un chemin partiel en considérant des ressources de visite binaires qui indiquent s'il est possible ou non de visiter un client pour un chemin partiel. Tel que décrit dans les sections suivantes, les règles d'élimination et de dominance (comparaison des différentes étiquettes) sont significativement influencées par la présence de ces ressources de visite.

4.2.11 Règles de dominance en absence de ressources de visite

Lorsqu'on ne considère par de ressources de visite binaires, une étiquette non Pareto optimale n'est pas nécessairement inutile. En effet, il est possible que l'extension de certains chemins non Pareto optimaux mène à des chemins Pareto optimaux à cause des contraintes d'élimination de cycles. Par exemple, s'il existe deux étiquettes L_v^1 et L_v^2 associées à des chemins partiels différents de u à v dans $\mathcal{GS}$ ($v \neq os'$ si $u = os$) et que L_v^1 est Pareto optimale alors que L_v^2 ne l'est pas, il est possible que la prolongation de L_v^2 au noeud $t \neq os'$ mène à un chemin Pareto-optimal de u à t si L_v^1 ne peut être prolongée à ce nœud à cause de contraintes d'élimination de cycles. On ne peut donc pas se contenter de considérer uniquement les éti-

quettes Pareto optimales de u à v.

Il devient donc nécessaire de définir des chemins *utiles* de u à v, c.-à-d. ceux qui peuvent permettre d'obtenir un chemin Pareto-optimal de u à v . Tout d'abord, si $L_v^0, L_v^2 \in \mathcal{F}_{\mathcal{GS}}(os, v) \cap \mathcal{S}_{\mathcal{GS}}$ sont deux étiquettes associées au noeud $v \in C(j)$ et que L_v^0 est Pareto optimale alors que L_v^2 ne l'est pas, l'étiquette L_v^2 est dominée par l'étiquette L_v^0 et on dénote $L_v^0 \prec_{dom} L_v^2$. On peut ensuite dire que L_v^0 est utile si la condition suivante est respectée :

Definition 2. *Chemins utiles de i à j (Irnich et Villeneuve, 2004)*

$$\mathcal{E}(L_v^0) \not\subseteq \bigcup_{L_v^2 \in \mathcal{F}_{\mathcal{GS}}(u,v) \cap \mathcal{S}_{\mathcal{GS}} \,:\, L_v^2 \prec_{dom} L_v^0} \mathcal{E}(L_v^2) \tag{4.15}$$

Ainsi, toute étiquette Pareto optimale L_v^0 de u à v est utile, car $\nexists L_v^2 \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S}_{\mathcal{GS}} : L_v^2 \prec_{dom} L_v^0 \Rightarrow \mathcal{E}(L_v^0) \not\subseteq \emptyset$. Par contre, il est également possible qu'une étiquette $L_v^1 \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S}$ non Pareto optimale dominée par $L_v^2 \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S}_{\mathcal{GS}}$ soit utile si $\exists \epsilon \in \mathcal{E}(L_v^1) : L_v^1 \oplus_{\mathcal{G}} \epsilon \in \mathcal{F}_{\mathcal{GS}}(u, os') \cap \mathcal{S}_{\mathcal{GS}}$ et $L_v^2 \oplus_{\mathcal{G}} \epsilon \notin \mathcal{F}_{\mathcal{GS}}(u, os') \cap \mathcal{S}_{\mathcal{GS}}$. En d'autres mots, L_v^0 est utile même si elle est dominée par L_v^2, si au moins une de ses extensions n'est pas une extension réalisable pour L_v^2.

Si on résout un SPPRC-2-cyc sans ressources de visite, il n'est donc pas suffisant de considérer uniquement les étiquettes Pareto optimales à chacun des nœuds. Il est toutefois possible de limiter le nombre d'étiquettes utiles, car les conditions suivantes sont suffisantes pour éliminer une étiquette $L_v^3 \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S}_{\mathcal{GS}}$, si $\exists L_v^1, L_v^2 \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées.

Definition 3. *Règle de dominance générique avec élimination de 2 cycles sans ressources de visite*

$$\bar{v}(L_v^1) = \bar{v}(L_v^2) = \bar{v}(L_v^3) \tag{4.16}$$

$$r^{dem}(L_v^1) \leq r^{dem}(L_v^3) \text{ et } r^{dem}(L_v^2) \leq r^{dem}(L_v^3) \tag{4.17}$$

$$pred(L_v^1)) \neq pred(L_v^2) \neq pred(L_v^3) \tag{4.18}$$

$$\bar{c}(L_v^1) \leq \bar{c}(L_v^3) \text{ et } \bar{c}(L_v^2) \leq \bar{c}(L_v^3) \tag{4.19}$$

Sommairement, on conserve la meilleure et la deuxième meilleure étiquette ayant des prédécesseurs différents plutôt que de conserver uniquement la meilleure comme on le ferait

pour un SPPRC. En pratique, on peut également éliminer l'étiquette L_v^0 si cette dernière est dominée par une autre étiquette ayant le même prédécesseur qu'elle ou si L_v^0 ne peut être étendue au prédécesseur de l'étiquette dominante à cause de contraintes de ressources. Il est également possible de conserver uniquement la meilleure étiquette si cette dernière est fortement dominante. Ceci sera le cas si l'étiquette ne peut retourner à son prédécesseur à cause des contraintes de ressources (voir (Larsen, 1999)).

On note par ailleurs que la condition (4.16) implique (4.17) dans le cas de l'algorithme avant, car $\bar{v}(L_v^1) = \bar{v}(L_v^2) = \bar{v}(L_v^3) = \mathscr{V}(\mu_{Client(v)}, Client(v))$. On peut donc se contenter de vérifier (4.16), (4.18) et (4.19). Ceci n'est toutefois pas le cas pour l'algorithme arrière.

4.2.12 Règles de dominance en présence de ressources de visite

Comme mentionné précédemment, l'élimination de 2-cycles peut être effectuée en conservant au plus deux fois plus d'étiquettes qu'un SPPRC standard (pour chaque nœud, il suffit de conserver la meilleure et la seconde meilleure étiquette). Toutefois, lorsqu'on cherche à éliminer des cycles de taille $\kappa > 2$. On doit donc considérer $\mathcal{E}(L)$, l'ensemble des extensions possibles pour une étiquette donnée. Tel que décrit dans (Irnich et Villeneuve, 2004), il est possible de codifier $\mathcal{E}(L)$ pour toute étiquette L. Toutefois, leur technique nécessite des structures de données complexes et la complexité de calcul croit très rapidement lorsqu'on augmente la taille des cycles interdits. Afin d'éviter d'avoir à recourir à de telles structures, on choisit d'ajouter des ressources de visite binaires indiquant si un chemin partiel peut visiter un client ou non.

Lorsqu'on ajoute des ressources de visite binaires dans le cadre d'un ESPPRC, on considère explicitement les nœuds que l'on ne peut revisiter et $\mathcal{F}_{\mathcal{GS}}(os, os') \subseteq \mathcal{S}_{\mathcal{GS}}$, car $\mathcal{S}_{\mathcal{GS}} = \{$ étiquettes associées aux routes élémentaires dans $\mathcal{G}$ $\}$ et $\mathcal{F}_{\mathcal{GS}}(os, os') = \{$ étiquettes réalisables par rapport aux contraintes sur les ressources de visite de demande cumulée inférieure ou égale à Q et correspondant à des routes élémentaires dans $\mathcal{G}\}$. Par contre, si on tente de résoudre un ESPPRC sans ressources de visite, on a $\mathcal{F}_{\mathcal{GS}}(os, os') \not\subseteq \mathcal{S}_{\mathcal{GS}}$, car $\mathcal{F}_{\mathcal{GS}}(os, os') = \{$ étiquettes associées aux routes de os à os' de demande cumulée inférieure ou égale à $Q\}$ et $\mathcal{S}_{\mathcal{GS}} = \{$ étiquettes associées aux routes de os à os' correspondant à des routes élémentaire dans $\mathcal{G}$ $\}$. Contrairement au SPPRC-2-cyc sans ressources de visite, chaque étiquette Pareto optimale du ESPPRC avec ressources de visite correspond ainsi à un chemin Pareto-optimal. Les chemins non Pareto optimaux sont donc inutiles et il est suffisant de considérer uniquement les chemins Pareto optimaux. Dans ce cas, toute extension réalisable Pareto optimale sera nécessairement issue d'un étiquette Pareto optimale. En effet, si on considère

$\mathcal{RES} = \{\text{dem}, \text{visit}_1, ..., \text{visit}_n\}$, on note que pour deux étiquettes L_v^1 et L_v^2 :

$$r^s(L_v^1) \leq r^s(L_v^2)\ \forall s \in \mathcal{RES} \Rightarrow \mathcal{E}(L_v^2) \subseteq \mathcal{E}(L_v^1) \tag{4.20}$$

Avec des ressources de visite, si l'étiquette L_v^1 n'a pas visité plus de clients que L_v^2 et n'a pas accumulé plus de demande, alors toute extension réalisable de L_v^2 le sera également pour L_v^1. Par contre,

$$r^s(L_v^1) \leq r^s(L_v^2)\ \forall s \in \mathcal{RES} \not\Leftarrow \mathcal{E}(L_v^2) \subseteq \mathcal{E}(L_v^1) \tag{4.21}$$

Par exemple, si on considère deux étiquettes L_v^1 et L_v^2 associées à des chemins de la source à v réalisables dans le graphe $\mathcal{GS}$ et $\bar{v}(L_v^1) = \bar{v}(L_v^2) = v$. Si on suppose également que L_v^1 a déjà visité tous les clients et que L_v^2 a déjà visité tous les clients sauf le client i, mais que ce client est inaccessible (par exemple à cause d'une décision de branchement), on a alors $r^{\text{visit}_i}(L_v^2) < r^{\text{visit}_i}(L_v^1)$ et $r^{\text{dem}}(L_v^2) < r^{\text{dem}}(L_v^1)$, car la demande de tous les clients est strictement positive et l'étiquette L_v^1 a visité tous les clients de L_v^2 ainsi que le client i. Dans le cadre d'un ESPPRC, la seule extension réalisable pour ces deux étiquettes sera issue d'un arc du client actuel au puits et on aura $\mathcal{E}(L_v^2) = \mathcal{E}(L_v^1)$.

En ajoutant des ressources de visite, la règle de dominance suivante est donc suffisante et on peut éliminer l'étiquette $L_v^2 \in \mathcal{F}_{\mathcal{GS}}(u,v) \cap \mathcal{S}_{\mathcal{GS}}$ si $\exists L_v^1 \in \mathcal{F}(u,v)_{\mathcal{GS}} \cap \mathcal{S}_{\mathcal{GS}}$.

Definition 4. *Règle de dominance générique avec ressources de visite - SPPRC avec voisinage sans élimination de 2-cycles ou (E)SPPRC*

$$\bar{v}(L_v^1) = \bar{v}(L_v^2) \tag{4.22}$$
$$\bar{c}(L_v^1) \leq \bar{c}(L_v^2) \tag{4.23}$$
$$r^s(L_v^1) \leq r^s(L_v^2)\ \forall s \in \mathcal{RES} \tag{4.24}$$

L'ajout de contraintes d'élémentarité rend l'élimination d'étiquettes beaucoup plus difficile, car le nombre de critères à considérer augmente beaucoup lors de la comparaison d'étiquettes. Plutôt que d'avoir un maximum de $Q - E[\xi_i]$ chemins Pareto optimaux au client i, on en a maintenant un maximum de $(Q - E[\xi_i])2^n$; et ce pour tout client i. Le nombre d'étiquettes utiles croit donc extrêmement rapidement à chaque itération d'un ESPPRC.

Afin de trouver un compromis entre l'absence totale de ressources de visite pour le SPPRC et la création de $n = |\mathcal{N}|$ ressources binaires pour le ESPPRC, il est possible de considérer uniquement $\Delta(\mathcal{N}_i)$ ressources à chaque noeud i et d'empêcher les retour aux clients interdits. C'est ce que l'on fait lorsqu'on utilise les ng-routes et le SPPRC avec voisinage.

L'utilisation de chemins élémentaires implique l'élimination de 2-cycles. Toutefois, ce n'est pas le cas avec les ng-routes standard. La règle (4) ne serait donc pas utilisable si on veut combiner les ng-routes à l'élimination de 2-cycles et que les ressources de visite sont uniquement fixées en fonction des clients interdits par les ng-routes (en ne fixant pas explicitement la ressource de visite associée au prédécesseur à 1). Par contre, on peut combiner les ng-routes et l'élimination de 2-cycles en interdisant le retour aux prédécesseurs en fixant dynamiquement la ressource de visite associée aux clients prédécesseurs à l'aide de (4.9). La règle (4) devient alors suffisante pour éliminer des étiquettes, car les étiquettes utiles sont uniquement des étiquettes Pareto optimales.

Si la règle (4) n'est pas suffisante à cause de la consommation des ressources de visite (i.e. (4.22) et (4.23) sont respectées, mais la condition (4.24) ne l'est pas), alors on aura :

$$\bar{v}(L_v^1) = \bar{v}(L_v^2) \tag{4.25}$$
$$\bar{c}(L_v^1) \leq \bar{c}(L_v^3) \tag{4.26}$$
$$r^s(L_v^1) \not\leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \tag{4.27}$$

Cependant, il sera quand même possible d'effectuer de la dominance. On peut notamment éliminer l'étiquette $L_v^3 \in \mathcal{F}_{\mathcal{GS}}(u,v) \cap \mathcal{S}_{\mathcal{GS}}$, si $\exists L_v^1, L_v^2 \in \mathcal{F}_{\mathcal{GS}}(u,v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées.

Definition 5. *Règle de dominance générique avec ressources de visite - SPPRC avec voisinage et élimination de 2-cycles*

$$\bar{v}(L_v^1) = \bar{v}(L_v^2) = \bar{v}(L_v^3) \tag{4.28}$$
$$\bar{c}(L_v^1) \leq \bar{c}(L_v^3)\ \textit{et}\ \bar{c}(L_v^2) \leq \bar{c}(L_v^3) \tag{4.29}$$
$$r^s(L_v^1) \leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \setminus \{visit_j\}\ \textit{et}$$
$$r^s(L_v^2) \leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \setminus \{visit_i\} \tag{4.30}$$

où

$$Client(pred(L_v^3)) = h,\ Client(pred(L_v^2)) = i,\ Client(pred(L_v^1)) = j$$

Cette règle représente une combinaison de la règle pour l'élimination des 2-cycles et de celle avec ressources de visite. Elle est complémentaire à la règle (4) et s'applique lorsque cette dernière n'est pas utilisable. Bien qu'elle nécessite plus d'étiquettes pour effectuer la dominance, elle est un peu moins restrictive que (4), car elle n'effectue pas la dominance sur toutes les ressources de visite. Elle se base sur l'idée qu'il est possible d'éliminer l'étiquette L_v^3 à l'aide de L_v^1 et L_v^2 lorsque $\mathcal{E}(L_v^3) \subseteq \mathcal{E}(L_v^1) \cup \mathcal{E}(L_v^2)$ et que chaque extension dans $\mathcal{E}(L_v^1) \cup \mathcal{E}(L_v^2)$ correspond à une extension de $\mathcal{E}(L_v^3)$ ayant un coût réduit plus petit ou égal.

Contrairement au cas sans ressources de visite (règle (3)), il est inutile de considérer explicitement le prédécesseur des étiquettes, car ils sont déjà évalués lors de la mise à jour des ressources de visite à l'aide de (4.9). On note également qu'il est inutile de considérer la consommation de la ressource de visite associée au prédécesseur des étiquettes L_v^1 et L_v^2 (voir (4.30)). On considère les deux cas suivants :

1) Si (4.22) et (4.23) sont respectées par L_v^1 et L_v^3, que $r^s(L_v^1) \leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \backslash \{visit_j\}$ et $\boldsymbol{r^{visit_j}(L_v^3) = 1 = r^{visit_j}(L_v^1)}$, alors on a $r^s(L_v^1) \leq r^s(L_v^3) \forall s \in \mathcal{RES}$ et la règle (4) permet d'éliminer L_v^3. En d'autres mots, si L_v^3 ne peut être étendue au prédécesseur de L_v^1 et que L_v^1 a une consommation de ressource inférieure pour toutes les ressources, on sait que $\mathcal{E}(L_v^3) \subseteq \mathcal{E}(L_v^1)$ et que L_v^1 domine L_v^3. On peut appliquer le même raisonnement avec L_v^2 et L_v^3.

2) Si on suppose que (4.22) et (4.23) sont respectées par L_v^1 et L_v^3, que $r^s(L_v^1) \leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \setminus \{visit_j\}$ mais que $\boldsymbol{r^{visit_j}(L_v^3) = 0 < r^{visit_j}(L_v^1)}$, alors L_v^1 seule ne peut dominer L_v^3. Par contre, si $\exists L_v^2$ telle que (4.22) et (4.23) sont respectées par rapport à L_v^3 et $r^s(L_v^2) \leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \setminus \{visit_i\}$, alors on a $r^{visit_j}(L_v^2) \leq r^{visit_j}(L_v^3) = 0$. Il est donc possible d'étendre L_v^2 vers le client associé au prédécesseur de L_v^1. Similairement, on a $r^{visit_i}(L_v^1) \leq r^{visit_i}(L_v^3) = 0$ (si $r^{visit_i}(L_v^3) = 1$, alors on se reporte au cas 1 et L_v^2 domine L_v^3). Ainsi, on est également capable d'étendre L_v^1 vers le client associé au prédécesseur de L_v^2. Les extensions des étiquettes L_v^1 et L_v^2 couvrent donc toutes les extension de L_v^3 et cette dernière peut être éliminée.

4.2.13 Règles de dominance agrégée

Avec les règles de dominance énoncées précédemment, la dominance s'effectue uniquement lorsque deux chemins partiels dans $\mathcal{GS}$ on le même noeud terminal dans $\mathcal{GS}$. Toutefois,

on sait que chaque nœud dans $C(i) = \{\mathscr{V}(\mu, i) \in \mathcal{NS} | \mu \in \tilde{\mu}(i)\}$ représente un niveau de consommation de la ressource de demande cumulée lorsque le véhicule atteint le client i. De plus, on sait que si on applique un algorithme d'étiquetage sur $\mathcal{G}$, alors l'étape de dominance se produira à chaque fois qu'on considère des chemins partiels arrivant au même *client*. On peut donc raffiner les règles de dominance précédentes en les appliquant sur chaque ensemble $C(i)$ plutôt que sur les nœuds individuels.

L'étiquette $L_v^0 \in \mathcal{F}_{\mathcal{GS}}(os, v) \cap \mathcal{S}_{\mathcal{GS}}$ peut être éliminée si $\exists L_v^1, L_v^2 \in \mathcal{F}_{\mathcal{GS}}(os, v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées.

Definition 6. *Règle de dominance agrégée avec élimination de 2-cycles sans ressources de visite*

$$Client(\bar{v}(L_v^1)) = Client(\bar{v}(L_v^2)) = Client(\bar{v}(L_v^0)) \tag{4.31}$$
$$r^{dem}(L_v^1) \leq r^{dem}(L_v^0) \text{ et } r^{dem}(L_v^2) \leq r^{dem}(L_v^0) \tag{4.32}$$
$$Client(pred(L_v^1)) \neq Client(pred(L_v^2)) \neq Client(pred(L_v^0)) \tag{4.33}$$
$$\bar{c}(L_v^1) \leq \bar{c}(L_v^0) \text{ et } \bar{c}(L_v^2) \leq \bar{c}(L_v^0) \tag{4.34}$$

L'étiquette $L_v^0 \in \mathcal{F}_{\mathcal{GS}}(os, v) \cap \mathcal{S}_{\mathcal{GS}}$ peut quant à elle être éliminée si $\exists L_v^1 \in \mathcal{F}_{\mathcal{GS}}(os, v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées :

Definition 7. *Règle de dominance agrégée avec ressources de visite - SPPRC avec voisinage sans élimination de 2-cycles ou (E)SPPRC*

$$Client(\bar{v}(L_v^1)) = Client(\bar{v}(L_v^0)) \tag{4.35}$$
$$\bar{c}(L_v^1) \leq \bar{c}(L_v^0) \tag{4.36}$$
$$r^s(L_v^1) \leq r^s(L_v^0)\ \forall s \in \mathcal{RES} \tag{4.37}$$

Si on combine le SPPRC avec voisinage et l'élimination de 2-cycles, on peut également éliminer l'étiquette $L_v^3 \in \mathcal{F}_{\mathcal{GS}}(u, v) \cap \mathcal{S}_{\mathcal{GS}}$ si $\exists L_v^1, L_v^2 \in \mathcal{F}v(u, v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées.

Definition 8. *Règle de dominance agrégée avec ressources de visite - SPPRC avec voisinage et élimination de 2-cycles*

$$Client(\bar{v}(L_v^1)) = Client(\bar{v}(L_v^2)) = Client(\bar{v}(L_v^3)) \tag{4.38}$$

$$\bar{c}(L_v^1) \leq \bar{c}(L_v^3) \text{ et } \bar{c}(L_v^2) \leq \bar{c}(L_v^3) \tag{4.39}$$

$$r^s(L_v^1) \leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \setminus \{visit_j\} \text{ et} \tag{4.40}$$

$$r^s(L_v^2) \leq r^s(L_v^3)\ \forall s \in \mathcal{RES} \setminus \{visit_i\} \tag{4.41}$$

où

$$Client(pred(L_v^3)) = h,\ Client(pred(L_v^2)) = i,\ Client(pred(L_v^1)) = j$$

Proposition 1. *La règle de dominance agrégée est valide pour l'algorithme de fixation d'étiquettes « avant » à tout client $i \in \mathcal{N}$.*

Preuve 1. *Soit :*
$L_u^{fw,1}$ *et* $L_v^{fw,2}$*, deux étiquettes « avant » telles que :*
- $Client(\bar{v}(L_u^{fw,1})) = Client(\bar{v}(L_v^{fw,2})) = i$
- $\mu_1 = r^{dem}(L_u^{fw,1}) < r^{dem}(L_v^{fw,2}) = \mu_2$.
- $\bar{c}(L_u^{fw,1}) \leq \bar{c}(L_v^{fw,2})$

La proposition précédente est vérifiée si on peut montrer que toute extension réalisable de $L_u^{fw,2}$ sera également dominée par toute extension réalisable de $L_v^{fw,1}$ (selon l'utilisation ou l'absence de ressources de visite). Ceci sera vrai si on peut prouver la proposition suivante pour tout client $j \in \mathcal{N} \setminus \{i\}$:

Proposition 2. *Si le* client *j est atteignable grâce à un chemin dans $\mathcal{GS}$ partant du noeud résidant de $L_u^{fw,2}$, alors j l'est également à partir de $L_v^{fw,1}$ grâce à un chemin dans $\mathcal{GS}$ de coût plus petit ou égal.*

Preuve 2. *On suppose que j n'a pas encore été visité, car sinon j ne serait atteignable par aucune étiquette ayant i comme nœud final. En utilisant la structure de la récursion du problème de sommes partielles, on constate que si l'on est capable d'atteindre une somme de μ_1 avec k éléments n'incluant pas j, alors on peut forcément atteindre une somme de $\mu_1 + E[\xi_j]$ avec k éléments et l'élément j. On sait également que $\mu_2 + E[\xi_j] \leq Q$ et $\mathscr{V}(\mu_2 + E[\xi_j], j) \in \mathcal{NS}$, car j est atteignable à partir de $L_u^{fw,2}$. On en déduit que $\mu_1 + E[\xi_j] < Q$, car $\mu_1 < \mu_2$. On sait également que $\mu_1 + E[\xi_j] > E[\xi_j]$ où $E[\xi_j]$ représente le plus petit niveau de consommation de capacité possible au client j, car $0 < E[\xi_i] \leq \mu_1$ et $E[\xi_i] \in \mathbb{N}$. On a donc $\mu_1 + E[\xi_j] \in \tilde{\mu}(i)$ et le nœud $\mathscr{V}(\mu_1 + E[\xi_j], j)$ doit exister (voir la figure 4.2). Comme on peut appliquer ce raisonnement pour tout nœud atteignable, il s'ensuit que toute extension réalisable de $L_v^{fw,2}$*

le sera également pour $L_u^{fw,1}$ *et on a* $\mathcal{E}^{fw}(L_v^{fw,2}) \subseteq \mathcal{E}^{fw}(L_u^{fw,1})$. *En d'autres mots,* $L_u^{fw,1}$ *peut visiter tous les mêmes clients que* $L_v^{fw,2}$ *(on considère une extension comme un ensemble de clients et non comme un ensemble de noeuds dans* $\mathcal{GS}$*).*

On sait que le coût de l'arc $(\mathscr{V}(\mu_1, i), \mathscr{V}(\mu_1 + E[\xi_j], j))$ *est inférieur au coût de l'arc* $(\mathscr{V}(\mu_2, i), \mathscr{V}(\mu_2 + E[\xi_j], j))$, *car* $\mu_1 + E[\xi_j] < \mu_2 + E[\xi_j]$ *et que* $EFC(\mu)$ *est croissant selon* μ *(voir la preuve en annexe).*

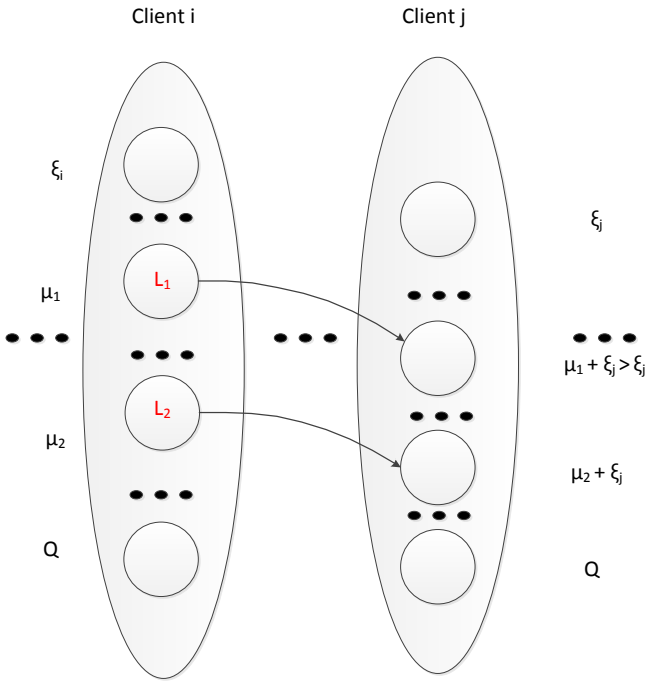

Figure 4.2 Validité de la dominance agrégé sur l'algorithme « avant »

Il est important de noter que cette règle est assurée d'être valide seulement si tous les arcs reliant initialement i à j existent toujours. Un problème pourrait notamment survenir si certaines règles de branchement avaient enlevé l'arc $(\mathscr{V}(\mu_1, i), \mathscr{V}(\mu_1 + E[\xi_j], j)$ du graphe

$\mathcal{GS}$.

Afin d'implanter cette nouvelle règle, il faut s'assurer que l'on compare les étiquettes associées à tout nœud $\mathscr{V}(\mu, i) \in C(i)$ avec toutes les étiquettes pertinentes associées aux nœuds dans $C(i)$. Comme une étiquette avec une plus grande consommation de ressource ne peut en dominer une autre avec une plus petite consommation, il s'ensuit qu'on peut se limiter à comparer uniquement une étiquette avec les étiquettes associées aux nœuds ayant une demande cumulée inférieure et étant associées au même client.

Il serait possible d'appliquer cette règle de dominance une fois tous les nœuds de $C(i)$ considérés par l'algorithme d'étiquetage (algorithme 3). Toutefois, ceci signifierait que l'on pourrait étendre des étiquettes inutiles à d'autres clients si on considère d'autres clients avant d'appliquer la règle de dominance aux nœuds dans $C(i)$; ce qui serait inefficace. La boucle 6 permet de contourner cette difficulté. En ajoutant des arcs artificiels entre les nœuds associés à un client donné et en sauvegardant cet ordre topologique à la ligne 12, on s'assure de pouvoir appliquer la règle de dominance à chaque itération de 17 si on compare les étiquettes à toutes les autres étiquettes ayant une demande espérée cumulée plus petite ou égale.

Proposition 3. *La règle de dominance agrégée n'est pas applicable lorsqu'on utilise un algorithme de fixation d'étiquettes qui progresse vers l'arrière.*

Preuve 3. *Le cas arrière pose problème, car une étiquette qui dominerait normalement ne peut pas nécessairement être étendue au même prédécesseur que l'étiquette dominée.*

Soit :
$L_u^{bw,1}$ *et* $L_v^{bw,2}$ *: deux étiquettes arrières avec*
- $Client(\bar{v}(L_u^{bw,1})) = Client(\bar{v}(L_v^{bw,2})) = j$
- $\mu_1 = r^{dem}(L_u^{bw,1}) < r^{dem}(L_v^{bw,2}) = \mu_2$
- $\bar{c}(L_u^{bw,1}) \leq \bar{c}(L_v^{bw,2})$.

Il n'est pas évident que toute extension arrière réalisable de $L_v^{bw,2}$ *soit dominée par une extension de* $L_u^{bw,1}$*. La figure 4.3 illustre bien la situation.*

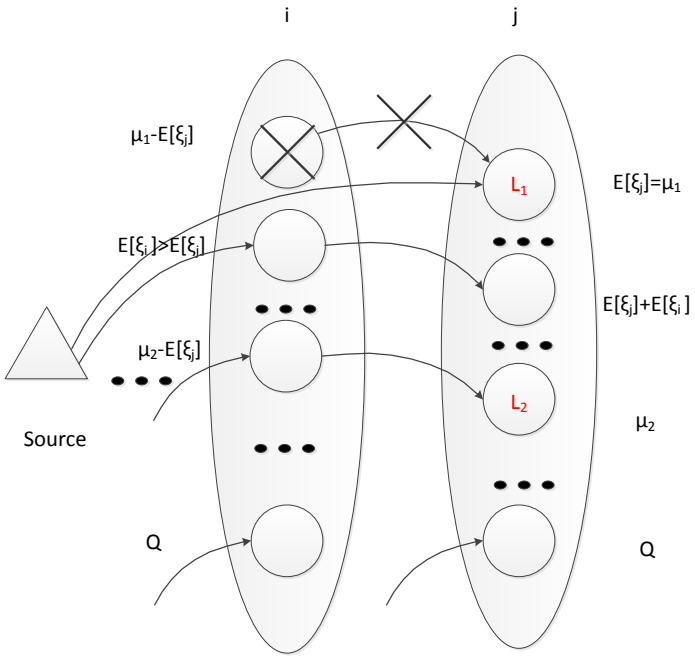

Figure 4.3 Problèmes avec dominance agrégée et progression arrière

En effet, si $\mu_1 = r^{dem}(L_u^{bw,1}) = E[\xi_j] < E[\xi_i]$ *et que* $E[\xi_i] + E[\xi_j] < r^{dem}(L_v^{bw,2}) = \mu_2 \leq Q$, *alors on sait que le nœud* $\mathscr{V}(\mu_2 - E[\xi_j])$ *existe, car* $E[\xi_i] < \mu_2 - E[\xi_j] \leq Q - E[\xi_j]$. *On a donc* $\mathscr{V}(\mu_2 - E[\xi_j]) \in \tilde{\mu}(j)$ *et on sait qu'il est possible d'étendre* $L_v^{bw,2}$ *directement vers i à l'aide de l'arc* $\mathscr{V}(\mu_2 - E[\xi_j], i), \mathscr{V}(\mu_2, j))$.

Par contre, l'étiquette $L_u^{bw,1}$ *ne peut être étendue au nœud i, car l'arc* $(\mathscr{V}(\mu_1 - E[\xi_j], i), \mathscr{V}(\mu_1, j))$ *n'existe pas. En effet, on sait que* $\mu > 0\ \forall \mathscr{V}(\mu, i) \in C(i)\ \forall i \in \mathcal{N}$. *Or, on a* $\mu_1 - E[\xi_j] = 0$. *Le nœud* $\mathscr{V}(\mu_1, i)$ *ne peut donc pas exister et il est impossible de garantir que toutes les extensions réalisables d'une étiquette ayant un vecteur de consommation plus grand soient également réalisables pour une étiquette ayant un plus petit vecteur de consommation. On ne peut donc pas utiliser la règle de dominance agrégée avec l'algorithme arrière.*

Intuitivement, un chemin partiel arrière dont le noeud terminal a une plus grande demande cumulée espérée indique que ce chemin pourra visiter plusieurs clients lorsqu'on l'étend vers la source. On peut effectivement montrer que pour deux étiquettes $L_u^{bw,1}, L_v^{bw,2}$ telles que $u = \mathscr{V}(\mu_1, i)$ et $v = \mathscr{V}(\mu_2, i)$ associées au client i et $\mu_1 < \mu_2$, alors $L_v^{bw,2}$ peut visiter au moins tous les clients que $L_u^{bw,i}$ peut visiter. Toutefois, on considère le coût total d'un chemin partiel et ce dernier prend en compte le coût espéré d'échec. Or, le coût espéré d'échec est croissant selon le demande cumulée. Ainsi, on ne peut appliquer de dominance arrière agrégée.

Il est donc nécessaire d'appliquer les règles de dominance standard non agrégées (règle 4, 3 et 5) sur les étiquettes lors de l'application de l'algorithme arrière. Toutefois, ceci implique que beaucoup moins d'étiquettes seront éliminées et que le nombre d'étiquettes considérées augmentera considérablement. Ceci pourrait effacer les avantages initiaux de la bidirectionalité.

4.2.14 Relation entre la valeur de la borne inférieure et le temps total de résolution selon le sous-problème

Bien que la permission de cycles permette de réduire le temps de résolution du sous-problème, cette relaxation est susceptible d'augmenter le temps total pris par l'algorithme de BCP. En effet, la borne inférieure produite par la relaxation linéaire de PM sera inférieure si on utilise un sous-problème de plus court chemin non élémentaire. Il sera ainsi plus difficile d'éliminer des nœuds de branchement lorsque la valeur optimale de la relaxation linéaire est plus élevée que la meilleure solution entière réalisable. Ceci augmentera la taille de l'arbre de branchement et pourrait ainsi augmenter le temps total de résolution.

Le compromis entre la valeur de la borne inférieure et le temps requis pour la résolution du sous-problème est bien illustré dans les figures 4.4, 4.5 et 4.6. Même si on augmente légèrement la qualité de la borne inférieure obtenue par la résolution de la relaxation linéaire du problème maître avec un sous-problème élémentaire, ceci engendre une augmentation considérable du temps de calcul.

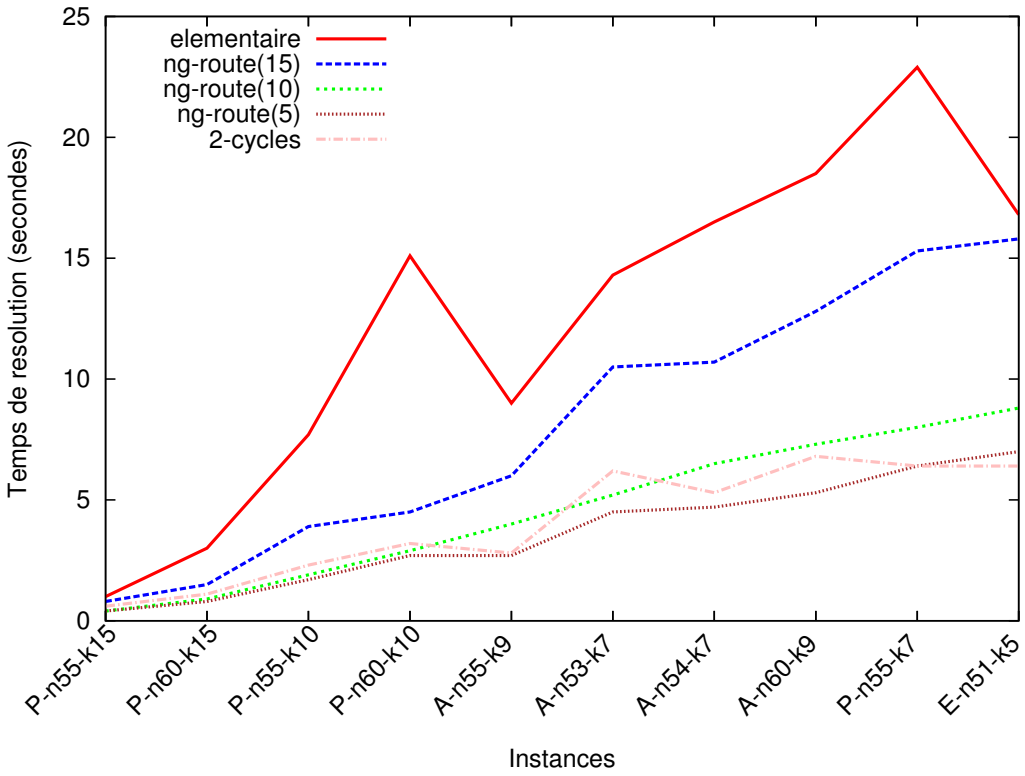

Figure 4.4 Meilleurs temps de calcul pour la relaxation linéaire au nœud racine selon les différentes relaxations du sous-problème

Les courbes indiquent les temps de résolution pour la relaxation linéaire du problème maître au nœud racine pour des instances sélectionnées. Elles représentent respectivement le temps pris avec le sous-problème élémentaire, le sous-problème avec ng-routes dont la cardinalité du voisinage est de 15, 10 et 5 clients et le sous-problème avec élimination de 2-cycles. L'élimination des 2-cycles n'est *pas* combinée aux ng-routes.

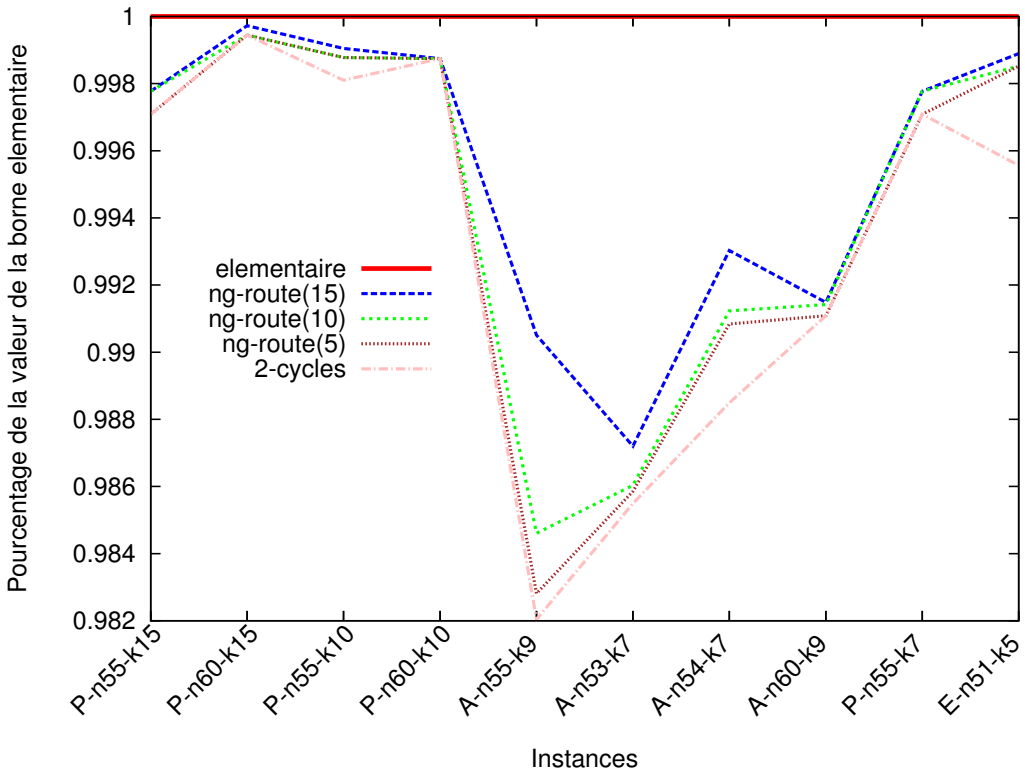

Figure 4.5 Rapport entre la borne inférieure des différentes relaxations et celle obtenue avec le sous-problème élémentaire initial

Il est important de mettre en perspective les valeurs présentées dans la figure 4.5 en tenant compte de l'échelle utilisée. On constate effectivement que les bornes inférieures sont toutes très similaires.

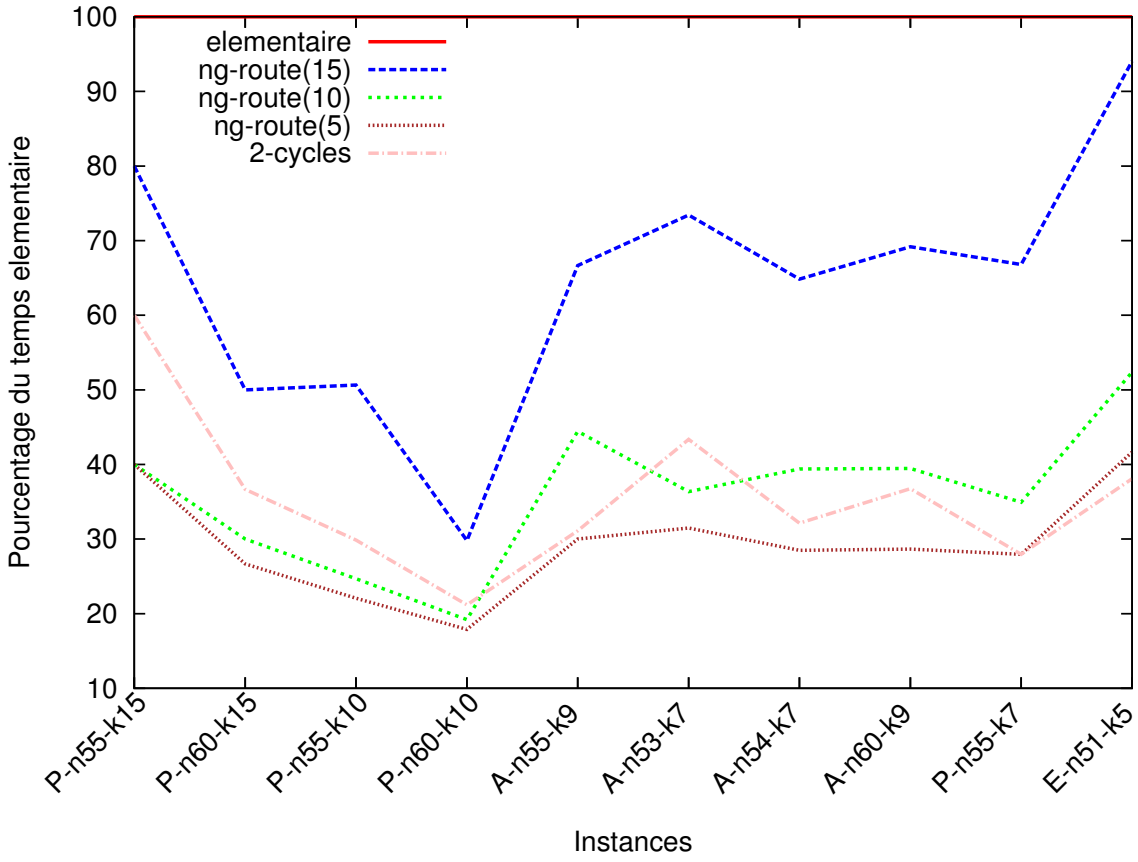

Figure 4.6 Rapport entre le temps de calcul des différentes relaxations et celui obtenu avec le sous-problème élémentaire initial

Bien que la figure 4.5 indique que les valeurs des bornes inférieures sont très rapprochées, on voit que les temps de calcul varient considérablement d'un type de sous-problème à l'autre. On note également que l'augmentation de $\Delta(\mathcal{N}_i)$ rapproche le temps de calcul avec ng-routes des résultats avec routes élémentaires. Pour certaines instances, il est toutefois possible d'obtenir une borne inférieure quasi-identique en considérant les ng-routes et les routes élémentaires, mais en ayant un temps de calcul considérablement inférieur avec les ng-routes. L'instance P-n60-k10 est un bon exemple, bien que la valeur de la borne inférieure soit 99.9% de celle du problème élémentaire, le temps de calcul n'est que de 29.8% du temps nécessaire pour résoudre le ESPPRC.

Ces résultats concernent uniquement la borne inférieure obtenue au nœud racine. Afin d'obtenir un portrait plus complet, il est préférable de se référer aux résultats finaux présentés à la section 5.2. Toutefois, les graphiques précédents illustrent bien les avantages potentiels d'utiliser les différents sous-problèmes dans certaines situations. Par exemple, certaines ins-

tances peuvent être résolues directement au nœud racine grâce à la borne inférieure élevée du sous-problème élémentaire. Dans d'autres cas où il est trop coûteux d'assurer l'élémentarité des routes, les ng-routes permettent d'établir un compromis efficace entre temps de calcul et qualité de la borne inférieure.

4.3 Algorithme de recherche taboue

Bien que le SPPRC avec voisinage et élimination de 2-cycles et le ESPPRC procurent des bornes inférieures plus fortes que ceux d'un SPPRC standard, ces problèmes sont très coûteux en termes de temps de calcul ; particulièrement le ESPPRC qui est $\mathcal{NP}$-difficile. Il est donc utile d'utiliser ces résolutions exactes uniquement vers la fin d'une itération de l'algorithme de génération de colonnes, c.-à.-.d. lorsqu'il reste peu de colonnes de coût réduit négatif à générer. L'ajout préliminaire de nombreuses colonnes de coût réduit négatif est susceptible d'accélérer la résolution du problème d'étiquetage exact. En effet, l'introduction de routes de faible coût devrait réduire la valeur des variables duales, ce qui devrait favoriser l'élimination de cycles et accélérer l'algorithme d'étiquetage. En utilisant l'algorithme exact uniquement lors des dernières itérations de l'algorithme de génération de colonnes, il est ainsi possible réduire le temps de calcul tout en préservant l'optimalité de la solution.

Nous adoptons cette stratégie en exploitant une recherche taboue qui permet de générer rapidement plusieurs colonnes de coût réduit négatif dans les premières itérations de l'algorithme de génération de colonnes. Cette méta-heuristique est non seulement plus rapide que les algorithmes de plus court chemin avec élimination de cycles, mais elle permet de générer plusieurs chemins réalisables de coût réduit négatif à chaque itération de l'algorithme de génération de colonnes.

L'algorithme tabou procède itérativement en considérant à chaque itération une route réalisable existante à modifier afin de créer des routes de coût réduit négatif. L'algorithme débute en utilisant des chemins associés aux variables en base dans la solution $\underline{\lambda}_k^l$ de PMR_k^l. Ces chemins représentent de bons candidats pour générer de nouveaux chemins de coût réduit négatif, car leurs variables correspondantes ont un coût réduit nul.

Pour générer de nouvelles routes, on tente successivement de supprimer un à un les clients étant dans la route actuellement considérée puis d'insérer les clients non visités par cette route. On permet seulement les mouvements non tabous, sauf dans le cas où la route que

l'on obtiendrait a un coût réduit strictement inférieur au coût de la meilleure route trouvée depuis le début de l'itération de génération de colonnes.

Au cours de ce processus d'évaluation, l'algorithme crée un ensemble de routes et il identifie la meilleure parmi celle-ci, c.-à-d. celle ayant le plus petit coût réduit. Même si cette dernière a un coût réduit plus grand ou égal à zéro ou plus grand ou égal au coût réduit de la meilleure solution trouvée jusque là, on la considère à la prochaine itération comme la route sur laquelle appliquer les opérateurs d'ajout et de suppression. Toute route valide engendrée par l'évaluation d'un opérateur est ajoutée au problème maître si elle a un coût réduit négatif. En effectuant ces opérations successives sur la meilleure route obtenue à partir d'un colonne en base, on procède à une intensification du processus de recherche.

On interdit ensuite l'inverse du mouvement qui a conduit à cette route réalisable (s'il existe) pour un certain nombre d'itérations en l'insérant dans une liste taboue. Par exemple, si l'insertion du client i mène au chemin p^* de coût réduit négatif minimal, alors on interdit sa suppression pour un certain nombre d'itérations dans cette route. Ceci permet d'éviter de cycler et de retourner périodiquement aux mêmes solutions de petit coût réduit. La liste taboue permet également de s'échapper des minima locaux.

Pour éviter de passer trop de temps à considérer un chemin, on impose que l'on ne peut dépasser un certain nombre d'itérations par chemin considéré. Si on ne parvient pas à trouver de routes de coût réduit négatif à partir d'une variable en base avant d'atteindre ce critère d'arrêt, alors on passe à un variable en base différente. Ceci assure une bonne diversification de recherche du domaine de solutions. On procède ainsi jusqu'à ce qu'il n'y ait plus de colonnes de coût réduit négatif ou qu'un critère d'arrêt soit rencontré.

De façon générale, une méthode de recherche taboue est définie par le triplet (f, S, N) ainsi que la liste taboue T :

- f : la fonction mesurant la qualité d'une solution obtenue ;
- S : l'espace des solutions considérées ;
- N : le voisinage considéré ;
- T : la liste taboue qui définit un ensemble de mouvements interdits pendant un certain nombre d'itérations.

Afin de bien définir ces caractéristiques pour notre algorithme, on introduit la notation suivante :

- $\mathscr{P}_k^l(\underline{\lambda}_k^l) \subseteq \mathscr{P}_k^l$: l'ensemble des routes correspondant aux variables en base dans le solution $\underline{\lambda}_k^l$ actuelle de PMR_k^l à la k^e itération de l'algorithme de génération de colonnes au l^e nœud de branchement ;
- $\mathscr{P}_k^{l-}(\underline{\lambda}_k^l)$: l'ensemble des routes de coût réduit négatif identifiées au cours de l'algorithme à partir de routes associées aux variables en base dans PMR_k^l ;
- $\theta_j^p(z) = p'$: la route obtenue en appliquant l'opérateur θ_j^p consistant à ajouter le client j dans le chemin p à la position z ;
- $\phi_j^{p'}(z) = p$: la route obtenue en appliquant l'opérateur $\phi_j^{p'}$ consistant à enlever le client j se trouvant à la position z dans le chemin p' ;
- $Inverse(m_j^p) = \begin{cases} \phi_j^p & \text{si } m = \theta \\ \theta_j^p & \text{sinon} \end{cases} \quad \forall j \in \mathcal{N}, p \in \mathscr{P}_k^l$
  $Inverse$ représente donc la fonction qui permet d'obtenir l'inverse du mouvement m_j^p ;
- $client(z, p)$: la fonction qui retourne le client se trouvant à la position z dans la route p ;
- $\Upsilon(p, m)$: Si $m = \theta$ et que l'on cherche à ajouter un client, $\Upsilon(p, m)$ représente l'ensemble des clients n'étant pas dans la route p (c.-à-d. les clients $i \in \mathcal{N} \setminus p$). Si $m = \phi$ et que l'on veut enlever un client de la route, alors $\Upsilon(p, m)$ représente l'ensemble des clients dans la route p ;
- $valide(m, p, j, z)$: fonction qui indique si la route $m_j^p(z) = p'$ est valide. Une route est valide si et seulement si les conditions suivantes sont respectées :
  - Elle ne peut avoir déjà été générée par l'algorithme ;
  - Elle ne peut avoir une demande cumulée espérée strictement plus grande à la capacité d'un véhicule ;
  - Si le sous-problème est un ESPPRC, la route doit être élémentaire ; un client ne peut donc pas apparaître plus d'une fois dans un route ;
  - La route doit respecter toutes les décisions de branchement en vigueur (e.g. deux clients doivent se suivre ou il leur est interdit de se suivre).
- $\mathscr{P}_{p^*} = \{p' \in \bigcup_{m \in \{\phi,\theta\}} \bigcup_{j \in \Upsilon(p,m)} \bigcup_{z=1}^{|p^*|} \{m_j^{p^*}(z)\} : \bar{c}(p') < 0, m_j^{p^*} \notin T \wedge valide(m, m_j^{p^*}(z), j, z)\} \setminus \mathscr{P}_k^{l-}(\underline{\lambda}_k^l)$: l'ensemble des routes de coût réduit négatif valides pouvant être obtenues grâce à l'utilisation d'un mouvement non tabou avec la route p^* actuellement considérée. Ces routes peuvent uniquement être générées une fois par l'algorithme ;
- $\hat{m}_j^{p^*} = \underset{m_j^{p^*}(z) \in \mathscr{P}_{p^*}}{\operatorname{argmin}} \{\bar{c}(m_j^{p^*}(z))\}$: le mouvement valide non tabou permettant de générer la route de plus petit coût réduit à partir de la route actuelle p^* ;
- $maxCol$: le nombre maximal de colonnes pouvant être générées pour une itération de

l'algorithme de génération de colonnes ;
- $\Delta(T) = 10$: La taille de la liste taboue. Bien que certains auteurs ait démontré l'utilité d'utiliser une mémoire adaptative ainsi qu'une taille variant dynamiquement pour la liste taboue T, nous considérons une taille fixe pour les besoins de parcimonie.

Dans notre algorithme, on fixe donc :
- $f = \bar{c}(p)$: le coût réduit d'une route ;
- $S = \mathscr{P}^l$: l'ensemble des routes réalisables au l^e nœud de branchement ;
- $N = \mathscr{P}_{p^*}$: l'ensemble des routes réalisables pouvant être obtenues en appliquant un mouvement unique à la route p^* actuellement considérée ;
- T : l'ensemble des mouvements interdits où $|T| \leq \Delta(T) = 10$.

L'algorithme est décrit en détail à la page suivante. La procédure **ModifierGraph**($\mathcal{GS}, SP_{type}, \pi_k$) est identique à celle utilisée pour le problème de plus court chemin. On effectue donc la recherche taboue sur le graphe dont le coût des chemins est modifié en fonction des variables duales à cette itération de l'algorithme de génération de colonnes et les inégalités valides identifiées à ce nœud de branchement.

Les lignes 2 et 3 initialisent les variables aux bonnes valeurs. On fixe notamment le meilleur coût réduit à zéro, car le coût réduit de tous les chemins associées aux colonnes en base est également égal à zéro.

Les lignes 6 à 35 représentent la recherche d'un meilleur voisin pour le chemin associé à la colonne en base actuellement considérée. Cette recherche consiste à identifier une solution réalisable de plus petit coût réduit pouvant être obtenue en appliquant un opérateur de suppression ou d'ajout de client dans la route p^* actuellement considérée à une position appropriée dans la route.

La ligne 14 nous assure que cette solution sera réalisable alors que la ligne 15 garantit que l'on ne peut pas utiliser de mouvement tabou, sauf si le mouvement considéré amène une solution meilleure que toutes celles trouvées jusque là. Comme le meilleur coût est initialisé à zéro, le mouvement candidat doit forcément créer une colonne de coût réduit strictement négatif. Ce critère d'aspiration standard permet de s'assurer que l'on n'élimine pas de solutions prometteuse. Bien qu'il soit théoriquement possible de regénérer une route, on s'assure de ne pas ajouter les doublons au problème maître.

Les lignes 16 à 18 permettent d'identifier les chemins associés à des variables de coût réduit strictement négatif. Les colonnes et variables associées à ces chemins sont ajoutées au problème maître. La condition à la ligne 19 permet de conserver le meilleur mouvement pour une itération de recherche de meilleur voisin (lignes 6 à 35) alors que 24 nous permet de conserver le meilleur coût réduit trouvé à une itération de génération de colonnes donnée.

Le test à la ligne 29 nous assure que l'on continue avec un chemin valide. Si cette condition échoue, on sort de la boucle débutant à la ligne 6 et on passe à une autre colonne en base. 30 nous assure que l'on conserve toujours une liste de taille fixe égale à $\Delta(T) = 10$.

Algorithme 4: RechercheTaboue($\mathcal{GS}, S^l_{IP,k}, \underline{\lambda}^l_k, \pi_k$)

ModifierGraph($\mathcal{GS}, SP_{type}, \pi_k$)
$\bar{c}^{**} \leftarrow 0$
$\mathscr{P}^{l-}_k(\underline{\lambda}^l_k) \leftarrow T \leftarrow \emptyset$
pour chaque $p \in \mathscr{P}^l_k(\underline{\lambda}^l_k)$ **faire**
  $p^* \leftarrow p$
  tant que $|\mathscr{P}^{l-}_k(\underline{\lambda}^l_k)| < maxCol$ **et** $p^* \neq \emptyset$ **faire**
    $\bar{c}^* \leftarrow \infty$
    $\hat{m}^{p^*}_j \leftarrow \emptyset$
    pour chaque $m \in \{\theta, \phi\}$ **faire**
      pour chaque $j \in \Upsilon(p, m)$ **faire**
        pour chaque $z \in \{1, 2, ..., |p^*|\}$ **faire**
          si $m = \phi$ **alors** $j \leftarrow client(z, p)$
          $p' \leftarrow m^{p^*}_j(z)$
          si $\neg valide(m, p', j, z)$ **alors continue**
          si $m^{p^*}_j \notin T$ **ou** $\bar{c}(p') < \bar{c}^{**}$ **alors**
            si $\bar{c}(p') < 0$ **alors**
              $\mathscr{P}^{l-}_k(\underline{\lambda}^l_k) \leftarrow \mathscr{P}^{l-}_k(\underline{\lambda}^l_k) \cup p'$
            fin
            si $\bar{c}(p') < \bar{c}^*$ **alors**
              $\hat{m}^{p^*}_j \leftarrow m^{p^*}_j$
              $\bar{c}^* \leftarrow \bar{c}(p')$
              $p^* \leftarrow p'$
            fin
            $\bar{c}^{**} \leftarrow \min\{\bar{c}^*, \bar{c}^{**}\}$
          fin
        fin
      fin
    fin
    si $\hat{m}^{p^*}_j \neq \emptyset$ **alors**
      si $|T| > \Delta(T)$ **alors** $T.pop()$
      $T \leftarrow T \bigcup Inverse(\hat{m}^{p^*}_j)$
    sinon
      $p^* \leftarrow \emptyset$
    fin
  fin
fin
retourne $\mathscr{P}^{l-}_k(\underline{\lambda}^l_k)$

Comme $S = \mathscr{P}^l$, on considère seulement les routes qui sont réalisables par rapport aux règles de branchement imposées. Par exemple, si deux clients doivent être visités un à la suite de l'autre, on ne peut insérer de nouveau client entre eux ou supprimer l'un deux si l'autre se trouve dans la route. Lorsque combiné avec le sous-problème élémentaire, on vérifie également qu'un client n'est pas déjà dans la route lors de l'insertion. Lorsqu'on élimine les deux cycles, on doit s'assurer de vérifier les routes engendrées lors de l'insertion et la suppression de clients. Les figures 4.7 et 4.8 illustrent bien les vérifications qui doivent être effectuées dans ces deux cas. Cette vérification est toutefois réalisable en temps constant et est relativement facile à effectuer.

Bien qu'il ne soit pas nécessaire de procéder ainsi, notre algorithme tabou utilise le graphe espace-état $\mathcal{GS}$ pour évaluer le coût et la faisabilité de nouvelles routes. Cette approche a des avantages, comme la facilité d'obtenir le coût réduit des différents arcs rapidement. Toutefois, elle possède également des limitations importantes.

La figure 4.7 démontre notamment l'un des désavantages d'utiliser le graphe espace-état $\mathcal{GS}$ avec la méthode taboue lorsqu'on tente d'insérer un client dans une route. Étant donné la route $p_{\mathcal{G}}$ dans $\mathcal{G}$ dont la route correspondante dans $\mathcal{GS}$ est $p_{\mathcal{G}} = \{j_1 = os, j_2, ..., j_k = i, j_{k+1} = i+1, ..., j_{|p|} = os'\}$, on doit supprimer chaque arc $(\mathscr{V}(\mu^p_{j_z}, j_z), \mathscr{V}(\mu^p_{j_{z+1}}, j_{z+1}))$ où $k \leq z \leq |p| - 1$. Si on ajoute le client t après le client i, alors on doit également ajouter les arcs $(\mathscr{V}(\mu^p_i, i), \mathscr{V}(\mu^p_i + E[\xi_t], t))$, $(\mathscr{V}(\mu^p_i + E[\xi_t], t), \mathscr{V}(\mu^p_{i+1} + E[\xi_t], i+1)$ et $(\mathscr{V}(\mu^p_{j_z} + E[\xi_t], j_z), \mathscr{V}(\mu^p_{j_{z+1}} + E[\xi_t], j_{z+1}))$ pour $k+1 \leq z \leq |p| - 1$.

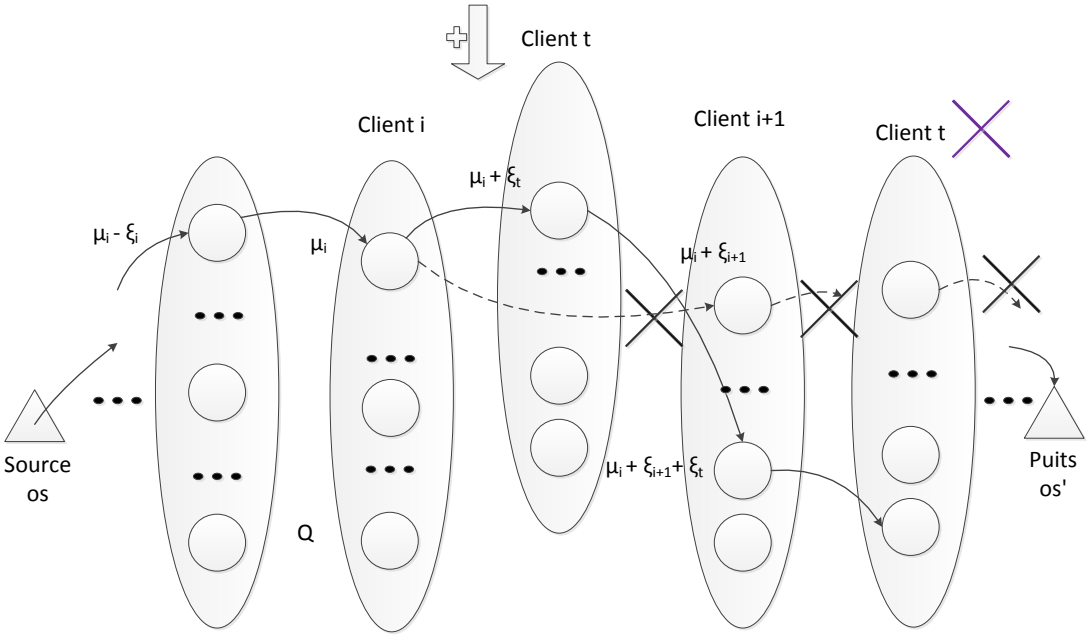

Figure 4.7 Difficulté lors de l'insertion de clients avec l'algorithme de recherche tabou

La figure 4.8 démontre les problèmes causés par la méthode taboue lorsqu'on tente de supprimer un client dans une route. Étant donné la route $p'_{\mathcal{G}}$ dans $\mathcal{G}$ dont la route correspondante dans $\mathcal{GS}$ est $p''_{\mathcal{G}} = \{j_1 = os, j_2, ..., j_{k-1} = i-1, j_k = i, j_{k+1} = t, j_{k+2} = i+1, ..., j_{|p'|} = os'\}$, on doit supprimer chaque arc $(\mathscr{V}(\mu^{p'}_{j_z}, j_z), \mathscr{V}(\mu^{p'}_{j_{z+1}}, j_{z+1}))$ où $k \leq z \leq |p'| - 1$. Si on enlève le client t, alors on doit également ajouter les arcs $(\mathscr{V}(\mu^{p'}_{i}, i), \mathscr{V}(\mu^{p'}_{i+1} - E[\xi_t], i+1))$ et $(\mathscr{V}(\mu^{p'}_{j_z} - E[\xi_t], j_z), \mathscr{V}(\mu^{p'}_{j_{z+1}} - E[\xi_t], j_{z+1}))$ pour $k+2 \leq z \leq |p'| - 1$.

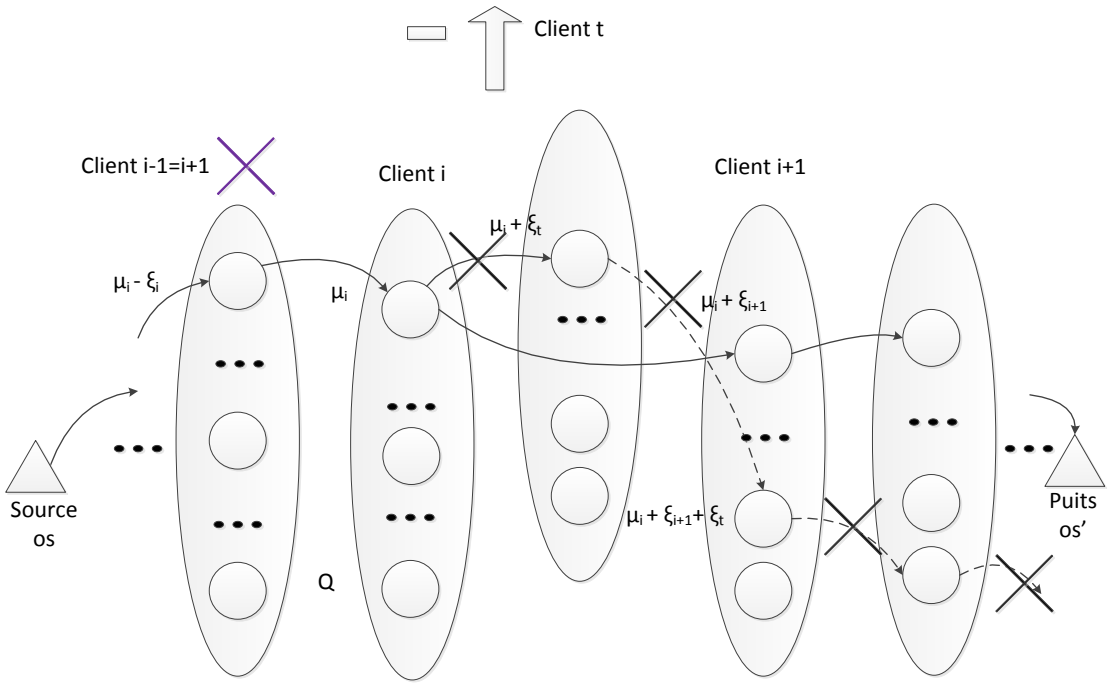

Figure 4.8 Difficulté lors de la suppression de clients avec l'algorithme de recherche tabou

4.4 Identification d'inégalités valides

Soit le problème DEP original formulé à partir du graphe $\mathcal{G}$:

$$\min_{x} \quad \sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N}'}\sum_{j\in\mathcal{N}'} c_{ij}x^v_{ij} + \mathcal{Q}(x) \tag{4.42}$$

$$(DEP) \qquad \text{s. à :} \quad \sum_{v\in\mathcal{V}}\sum_{(j,i)\in\delta^-(\{i\})} x^v_{ji} = 1, \quad \forall i \in \mathcal{N} \tag{4.43}$$

$$\sum_{(i,j)\in\delta^+(\{i\})} x^v_{ij} = \sum_{(j,i)\in\delta^-(\{i\})} x^v_{ji}, \quad \forall i \in \mathcal{N}, v \in \mathcal{V} \tag{4.44}$$

$$\sum_{(o,i)\in\delta^+(\{o\})} x^v_{oi} = \sum_{(i,o')\in\delta^-(\{o'\})} x^v_{io'} = 1, \quad \forall v \in \mathcal{V} \tag{4.45}$$

$$\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} E[\xi_j]x^v_{ij} \leq Q \quad \forall v \in \mathcal{V} \tag{4.46}$$

$$x^v_{ij} \in \{0,1\} \quad \forall i,j \in \mathcal{N}', \forall v \in \mathcal{V} \tag{4.47}$$

et le problème $DEPs$ équivalent, formulé sur le graphe $\mathcal{GS}$, :

$$\min_{\hat{x}} \quad \sum_{v\in\mathcal{V}}\sum_{a\in\mathcal{NS}}\sum_{b\in\mathcal{NS}} \hat{c}_{ab}\hat{x}^v_{ab} \tag{4.48}$$

$$(DEPs) \qquad \text{s. à} \quad \sum_{v\in\mathcal{V}}\sum_{(b,a)\in\delta^-(C(i))} \hat{x}^v_{ba} = 1, \quad \forall i \in \mathcal{N} \tag{4.49}$$

$$\sum_{(a,b)\in\delta^+(\{a\})} \hat{x}^v_{ab} = \sum_{(b,a)\in\delta^-(\{a\})} \hat{x}^v_{ba}, \quad \forall a \in \mathcal{NS}', \forall v \in \mathcal{V} \tag{4.50}$$

$$\sum_{(os,a)\in\delta^+(\{os\})} \hat{x}^v_{os,a} = \sum_{(a,os')\in\delta^-(\{os'\})} \hat{x}^v_{a,os'} = 1, \quad \forall v \in \mathcal{V} \tag{4.51}$$

$$\hat{x}^v_{ab} \in \{0,1\}, \quad \forall a,b \in \mathcal{NS}' \tag{4.52}$$

ainsi que le problème maître associé à ces deux problèmes :

$$\min_{\lambda} \quad \sum_{p\in\mathscr{P}} \hat{c}_p\lambda_p \tag{4.53}$$

$$(PM) \qquad \text{s. à :} \quad \sum_{p\in\mathscr{P}} \alpha_{ip}\lambda_p = 1, \quad \forall i \in \mathcal{N} \tag{4.54}$$

$$\lambda_p \in \{0,1\}, \ \forall p \in \mathscr{P} \tag{4.55}$$

où $\mathscr{P}$ représente l'ensemble des routes réalisables selon la consommation de ressources et n'importe quel type de sous-problème utilisé (élémentaire, avec élimination de cycles, etc.).

Il est possible de renforcer la relaxation linéaire de PM en ajoutant des inégalités valides

ou plans coupants. Ces inégalités sont respectées par toute solution entière réalisable, mais peuvent être violées par des solutions fractionnaires de la relaxation linéaire du problème maître. Ceci permet ainsi d'augmenter la borne inférieure produite par la résolution de la relaxation linéaire de PMR_k à chacune des itérations de l'algorithme de génération de colonnes (BP). Lorsque combiné à la méthode BP standard, l'ajout d'inégalités violées donne lieu à un algorithme de BCP.

On définit deux types d'inégalités valides pour la méthode de BCP : celles qui peuvent être introduites dans $DEPs$ et celles qui peuvent uniquement être introduites dans PM. On note que toute inégalité pouvant être introduite dans $DEPs$ peut également l'être dans DEP à l'aide de la transformation $\sum_{a \in C(i)} \sum_{b \in C(j)} \hat{x}^v_{ab} = x^v_{ij}$. On identifie les inégalités valides à partir du graphe $\mathcal{G}$ original, car la séparation est plus facile à réaliser lorsqu'on considère ce graphe réduit. On considère donc les inégalités dans DEP. Toutefois, comme on doit utiliser le graphe $\mathcal{GS}$ pour calculer le coût espéré d'échec d'une route lors de la résolution du sous-problème, on modifie le coût réduit des arcs dans $\mathcal{GS}$ à partir des inégalités dans DEP. Tous les arcs $(a, b) \in \mathcal{AS}$ correspondant à un arc $(i, j) \in \mathcal{A}$ faisant partie d'un contrainte dans DEP sont donc associés à la variable duale associée à cette contrainte dans DEP.

4.4.1 Inégalités valides dans $DEPs$ et DEP

L'ajout d'inégalités de ce type est simple, car ces dernières nécessitent uniquement de légères modifications au problème maître et son sous-problème. De plus, l'introduction de ces inégalités n'augmente pas la complexité de calcul du sous-problème. En effet, comme il est possible de les introduire dans DEP, ces contraintes peuvent être exprimées comme des combinaisons linéaires des variables de flot sur les arcs x^v_{ij}. Dans DEP, elles prennent la forme :

$$\sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}} \bar{\beta}_{ij} x^v_{ij} \leq \beta_0$$

où $\beta_0 \in \mathbb{N}$ représente un entier quelconque et $\bar{\beta}_{ij} \in \mathbb{N}$ représente le coefficient de la variable x^v_{ij} (la contribution de l'arc $(i, j) \in \mathcal{A}$ à la contrainte).

Par la suite, il est possible d'introduire ces inégalités dans PM grâce à la transformation :

$$\sum_{v \in \mathcal{V}} x^v_{ij} = \sum_{p \in \mathscr{P}} \alpha_{ijp} \lambda_p$$

où $\alpha_{ijp} \in \{0, 1\}$ est un paramètre indiquant si l'arc $(i, j) \in \mathcal{A}$ est emprunté par la route p.

On peut ensuite ajouter ces inégalités à PM en les exprimant comme :

$$\sum_{p \in \mathscr{P}} \beta_{ijp} \lambda_p \leq \beta_0$$

où $\beta_{ijp} = \bar{\beta}_{ijp} \alpha_{ijp}$ indique la contribution de l'arc $(i, j) \in \mathcal{A}$ à la contrainte ; si cette dernière fait partie de la route p correspondante dans $\mathcal{G}$.

Cette transformation est utilisée lors de la décomposition de Dantzig-Wolfe et peut, par exemple, être employée pour déterminer les coefficients des $|\mathcal{N}|$ contraintes de couverture des clients (contraintes liantes). On a $\sum_{(i,j) \in \delta^-(\{i\})} \alpha_{ijp} = \alpha_{ip}$ et $\sum_{p \in \mathscr{P}} \alpha_{ijp} \lambda_p = \sum_{v \in \mathcal{V}} x_{ij}^v$. On sait ainsi qu'il est possible d'exprimer ces contraintes en fonction des variables de flot sur les arcs x_{ij}^v ainsi qu'en fonction des variables de choix de routes λ_p. En effet, $\sum_{p \in \mathscr{P}} \alpha_{ip} \lambda_p = 1$ dans PM est équivalent à $\sum_{v \in \mathcal{V}} \sum_{(i,j) \in \delta^-(\{i\})} x_{ij}^v = 1$ dans DEP.

Pour notre problème particulier, nous avons considéré les inégalités du type $\sum_{p \in \mathscr{P}} \beta_{ijp} \lambda_p \leq \beta_0$ utilisées par le module CVRPSEP (Lysgaard., 2004). On considère deux types d'inégalités pouvant être exprimées en fonction des variables de flot sur les arcs.

◇ Coupes de capacité agrégées (« Aggregate Capacity Cuts » - ACC) :

Soit :
- $S \subset \mathcal{N}$: un sous-ensemble des clients tel que $2 \leq |S| \leq |\mathcal{N}|$;
- $r(S)$: le nombre minimal de véhicules nécessaires pour couvrir la demande espérée de tous les clients dans S.

$r(\mathcal{N})$ est obtenu en résolvant un problème de remplissage de sacs (« Bin packing problem » - BPP) où on considère tous les clients ($S = \mathcal{N}$) :

$$r(S) = \min_{s,t} \quad \sum_{v=1}^{|\mathcal{V}|} s_v \tag{4.56}$$

$$(BPP(S)) \qquad \text{s. à.} \quad \sum_{i \in S} s_{i,v} E[\xi_i] \leq Q t_v \quad \forall v \in \mathcal{V} \tag{4.57}$$

$$\sum_{v=1}^{|\mathcal{V}|} s_{i,v} = 1 \quad \forall i \in S \tag{4.58}$$

$$s_{i,v}, t_v \in \{0,1\} \quad \forall v \in \mathcal{V}, i \in S \tag{4.59}$$

t_v indique si on choisit le véhicule v et $s_{i,v}$ indique si la demande du client i est servie par le véhicule v. La contrainte (4.57) assure que l'on ne dépasse pas la capacité d'un véhicule alors que (4.58) impose qu'on assigne chaque client à exactement 1 véhicule. Afin d'éviter de résoudre ce problème $\mathcal{NP}$-difficile (Garey et Johnson, 1979), on utilise $k(S) = \left\lceil \frac{\sum_{i \in S} E[\xi_i]}{Q} \right\rceil$, une borne inférieure facilement calculable pour $r(S)$.

La contrainte de capacité agrégée s'énonce donc comme suit dans DEP :

$$\sum_{v \in \mathcal{V}} \sum_{(o,j) \in \mathcal{A}} x_{oj}^v \geq k(\mathcal{N}) \tag{4.60}$$

Dans PM, cette contrainte s'exprime comme :

$$\sum_{p \in \mathscr{P}} \lambda_p \geq k(\mathcal{N}) \tag{4.61}$$

Cette contrainte a une forme identique dans le cas du CVRP. Toutefois, le « poids » des items dans le BPP stochastique est donné par la demande espérée des clients plutôt que par la demande de ces derniers comme ce serait le cas pour le CVRP.

Pour la contrainte de capacité agrégée, on considère uniquement $BPP(S)$ avec $S = \mathcal{N}$. On ajoute donc uniquement une contrainte de capacité agrégée au problème maître. Par ailleurs, comme on connaît déjà la valeur de $k(\mathcal{N})$, on l'ajoute dans le problème maître restreint initial. Ceci permet d'augmenter rapidement la borne inférieure du problème à faible coût. Cette contrainte est toujours ajoutée de manière statique au problème maître au début de l'algorithme de BCP. On considère également les contraintes de capacité suivantes :

◇ Coupes de capacité (« Capacity cuts » - CC) :

$$\sum_{v\in\mathcal{V}}\sum_{(i,j)\in\delta^-(S)} x_{ij}^v \geq k(S) \quad \forall S \subseteq \mathcal{N}\ 2 \leq |S| \leq |\mathcal{N}| \tag{4.62}$$

Contrairement à la contrainte de capacité agrégée, seul un petit sous-ensemble de ces inégalités est ajouté à PM. Leur identification exhaustive serait prohibitive, car elles s'appliquent à tous les sous-ensembles de $\mathcal{N}$. Il existe donc un nombre exponentiel de telles contraintes. Par ailleurs, l'identification d'inégalités valides violées par des solutions fractionnaires nécessite la résolution d'un problème de séparation $\mathcal{NP}$-difficile. Par conséquent, on inclut dynamiquement uniquement les contraintes que l'algorithme de séparation du module CVRPSEP parvient à identifier heuristiquement et ayant une violation suffisamment élevée. Comme pour les ACC, on utilise $k(S)$ au lieu de $r(S)$.

Ce type de contraintes est facile à gérer, car on peut facilement modifier le coût réduit d'une route et donc la fonction objectif du sous-problème. On pose $\mathcal{H}_k$ comme l'ensemble des contrainte de capacité (CC) violées identifiées par l'algorithme de séparation à l'itération k, β_{ij}^f et σ_{ijk}^f représentent respectivement la contribution de l'arc (i,j) (s'il est emprunté par la route dans $\mathcal{G}$) et la valeur de la variable duale associée à la f^e contrainte de ce type à cette itération et φ comme la valeur de la variable duale associée à la contrainte de capacité agrégée.

Dans PM, ces contraintes s'expriment comme :

$$\sum_{p\in\mathscr{P}} \beta_{ij}^f \lambda_p \geq k^f(S),\ \forall f \in \mathcal{H}_k \tag{4.63}$$

Le coût réduit du sous-problème à l'itération k pour la route p empruntée par le véhicule v dans $\mathcal{GS}$ devient alors :

$$\bar{c}(p) = \sum_{\substack{(a,b)\in p\\ a\in C(i)\\ b\in C(j)}} (\hat{c}_{ab} - \pi_{jk} - \sum_{f\in\mathcal{H}_k} \beta_{ij}^f \sigma_{ijk}^f) - \varphi_k \tag{4.64}$$

4.4.2 Inégalités valides dans PM

Il est également possible d'ajouter des inégalités valides dans PM qui ne peuvent s'exprimer comme des combinaisons linéaires des variables de flot sur les arcs, x_{ij}^v. Bien qu'efficace

pour renforcer la relaxation linéaire du problème maître, ce type d'inégalités a seulement été récemment mis en pratique ; notamment dans les travaux de (Jepsen *et al.*, 2008). En effet, ces inégalités sont plus difficiles à traiter et il est nécessaire de modifier le sous-problème afin de calculer adéquatement le coût réduit des nouvelles routes. Elles doivent être introduites directement dans PM et prennent la forme :

$$\sum_{p \in \mathscr{P}} \gamma_p \lambda_p \leq \gamma_0$$

Pour notre problème particulier, nous avons considéré une seule famille d'inégalités de ce type :

◇ « Subset-row inequalities » - SRI
Ces inégalités prennent la forme :

$$\sum_{p \in \mathscr{P}} \left\lfloor \frac{1}{k} \sum_{i \in S} \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \frac{|S|}{k} \right\rfloor \tag{4.65}$$

où $0 < k \leq |S|$; $k \in \mathbb{N}$ et $S \subseteq \mathcal{N}$. Le coefficient α_{ip} a la même signification que α_{ip} dans PM, c.-à-d. qu'il indique le nombre de fois que le client i est visité par la route p.

On utilise spécifiquement celles avec $|S| = 3$ et $k = 2$:

$$\sum_{p \in \mathscr{P}} \left\lfloor \frac{1}{2} \sum_{i \in S} \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \frac{3}{2} \right\rfloor = 1 \tag{4.66}$$

Ces inégalités affirment que l'on peut choisir au plus 1 route couvrant 2 clients ou plus parmi un triplet de client ; et ce pour tout triplet. En effet, si $\sum_{i \in S} \alpha_{ip} \leq 1$ pour une route p et un triplet S donné, alors $\left\lfloor \frac{1}{2} \sum_{i \in S} \alpha_{ip} \right\rfloor = 0$. Cette route a un coefficient de 0 et n'est donc pas importante pour l'inégalité. Par contre, si $2 \leq \sum_{i \in S} \alpha_{ip} \leq 3$, alors $\left\lfloor \frac{1}{2} \sum_{i \in S} \alpha_{ip} \right\rfloor = 1$ et la route p sera prise en compte.

Si on définit $\mathcal{N}_p$ comme l'ensemble des clients couverts par la route p, $\mathcal{C} = \{S \subseteq \mathcal{N} : |S| = 3\}$ comme la famille des triplets de clients de $\mathcal{N}$ et $\mathscr{P}(S) = \{p \in \mathscr{P} : |\mathcal{N}_p \cap S| \geq 2\}$ comme l'ensemble des routes p couvrant au moins 2 clients dans un triplet $S \in \mathcal{C}$, ceci revient à (Baldacci *et al.*, 2007)

$$\sum_{p \in \mathscr{P}(S)} \lambda_p \leq 1 \; \forall \; S \in \mathcal{C} \tag{4.67}$$

Tel que mentionné dans (Jepsen *et al.*, 2008), les SRI peuvent être dérivées du graphe de conflit ; un graphe non dirigé dont les noeuds représentent les routes considérées à cette itération de l'algorithme de génération de colonnes (les routes p telles que $\lambda_p > 0$) et un arc entre deux noeuds indique que les deux routes sont en conflit, c.-à-d. qu'elles ont au moins un client en commun.

Lorsqu'on considère des routes *élémentaires*, les SRI identifiées empêchent de choisir au moins une arête dans le graphe de conflit. La solution fractionnaire actuelle n'est donc plus valide après l'introduction de la coupe. Toutefois, ceci n'est pas nécessairement le cas lorsqu'on considère les routes non élémentaires. Considérons l'exemple suivant de la figure 4.9 dans le graphe $\mathcal{G}$ original (où l'on omet les demandes et les demandes cumulées) :

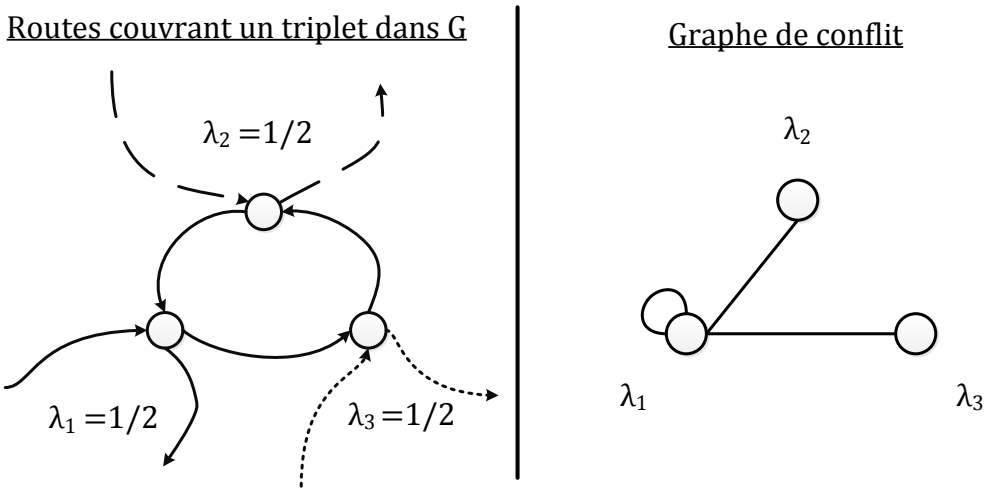

Figure 4.9 SRI avec routes non élémentaires

La SRI correspondante dans le problème maître sera :

$$\left\lfloor \frac{4}{2} \right\rfloor \lambda_1 + \left\lfloor \frac{1}{2} \right\rfloor \lambda_2 + \left\lfloor \frac{1}{2} \right\rfloor \lambda_3 \leq 1 \Rightarrow \lambda_1 \leq \frac{1}{2} \tag{4.68}$$

Cette solution fractionnaire ne violera donc pas cette inégalité et l'inégalité n'empêchera l'existence d'aucun arc dans le graphe de conflit. On note cependant que si un ensemble de routes élémentaires est empêché par l'introduction d'une SRI dans le PM utilisant un ESP-PRC comme sous-problème, cet ensemble le sera également dans le cas où l'on permet des cycles, car le coefficient α_{ip} a la même signification dans les deux sous-problèmes. Les SRI permettent donc de couper certaines solutions fractionnaires ; que l'on considère des routes élémentaires ou non. Néanmoins, il est possible qu'elles permettent d'obtenir une solution entière plus rapidement lorsqu'on considère des routes élémentaires.

Tout comme pour les CC, l'identification des SRI violées est $\mathcal{NP}$-difficile. Comme dans (Jepsen *et al.*, 2008) et (Desaulniers *et al.*, 2008), on énumère donc tous les sous-ensembles de clients de taille 3 en $O(n^3)$ et on identifie ensuite les inégalités valides violées par la solution fractionnaire actuelle.

4.4.3 Modification du sous-problème et de la règle de dominance

Pour un route p, le coefficient de la g^e SRI considérant l'ensemble S^g peut s'énoncer comme $\gamma_p^g = \left\lfloor \frac{|S^g \cap \mathcal{N}_p|}{2} \right\rfloor$. Si ν_k^g exprime la valeur de la variable duale associée à cette contrainte à l'itération $k-1$ de l'algorithme de génération de colonnes, alors on peut exprimer le coût réduit de la route p dans $\mathcal{GS}$ comme :

$$\bar{c}(p) = \sum_{\substack{(i,j)\in p \\ a\in C(i) \\ b\in C(j)}} (\hat{c}_{ab} - \pi_{jk} - \sum_{f\in\mathcal{H}} \beta_{ij}^f \sigma_{ijk}^f) - \varphi_k - \sum_{g\in\mathscr{G}} \gamma_p^g \nu_k^g \tag{4.69}$$

où $\mathscr{G}$ représente l'ensemble des SRI violées identifiées par la procédure de séparation.

Comme on ne peut décomposer les coefficients et les variables duales associées aux contraintes $\sum_{p\in\mathscr{P}} \gamma_p \lambda_p \leq \gamma_0$ par arcs, il est difficile de calculer adéquatement le coût réduit d'une route avec l'addition de ces contraintes. Pour ce faire, il faut augmenter la complexité de calcul de l'algorithme de plus court chemin et comptabiliser la contribution de chaque route aux différentes contraintes de ce type. L'algorithme de plus court chemin doit spécifi-

quement être modifié pour évaluer $|S^g \cap \mathcal{N}_p|$ pour la g^e SRI identifiée. On note que pour la g^e SRI, $\gamma_p^g \in \{0,1\}$, car $\gamma_p^g = \left\lfloor \frac{|S^g \cap \mathcal{N}_p|}{2} \right\rfloor$ et $0 \leq |S^g \cap \mathcal{N}_p| \leq 3$. On peut ainsi modifier les règles de dominance (6) et (7) comme suit. L'étiquette $L_v^0 \in \mathcal{F}_{\mathcal{GS}}(os,v) \cap \mathcal{S}_{\mathcal{GS}}$ peut être éliminée si $\exists L_v^1, L_v^2 \in \mathcal{F}_{\mathcal{GS}}(os,v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées.

Definition 9. *Règle de dominance agrégée avec élimination de 2-cycles sans ressources de visite et ajout de SRI*

$$Client(\bar{v}(L_v^1)) = Client(\bar{v}(L_v^2)) = Client(\bar{v}(L_v^0)) \tag{4.70}$$

$$r^{dem}(L_v^1) \leq r^{dem}(L_v^0) \text{ et } r^{dem}(L_v^2) \leq r^{dem}(L_v^0) \tag{4.71}$$

$$Client(pred(L_v^1)) \neq Client(pred(L_v^2)) \neq Client(pred(L_v^0)) \tag{4.72}$$

$$\bar{c}(L_v^1) - \sum_{g \in \mathscr{G}_{k,1,0}} \nu_k^g \leq \bar{c}(L_v^0) \text{ et } \bar{c}(L_v^2) - \sum_{g \in \mathscr{G}_{k,2,0}} \nu_k^g \leq \bar{c}(L_v^0) \tag{4.73}$$

$$\tag{4.74}$$

où $\mathscr{G}_{k,i,j} \subseteq \mathscr{G}_k$ représente un sous-ensemble des SRI identifiées telles que $\gamma_p^g = 1$ pour le chemin p dans $\mathcal{GS}$ représenté par L_v^i et $\gamma_{p'}^g = 0$ pour le chemin p' représenté par L_v^j où $p \neq p', j \neq i$. $\bar{c}(p) = \sum_{(i,j) \in p,\ a \in C(i), b \in C(j)} (\hat{c}_{ab} - \pi_{jk} - \sum_{f \in \mathcal{H}} \beta_{ij}^f \sigma_{ijk}^f) - \varphi_k$ représente le coût réduit sans prendre en compte les contraintes SRI.

Lorsqu'on ajoute des ressources de visite, l'étiquette $L_v^0 \in \mathcal{F}_{\mathcal{GS}}(os,v) \cap \mathcal{S}_{\mathcal{GS}}$ peut être éliminée si $\exists L_v^1 \in \mathcal{F}_{\mathcal{GS}}(os,v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées.

Definition 10. *Règle de dominance agrégée avec ressources de visite et ajout de SRI - SPPRC avec voisinage sans élimination de 2-cycles et ESPPRC*

$$Client(\bar{v}(L_v^1)) = Client(\bar{v}(L_v^0)) \tag{4.75}$$

$$\bar{c}(L_v^1) - \sum_{g \in \mathscr{G}_{1,0}} \nu_k^g \leq \bar{c}(L_v^0) \tag{4.76}$$

$$r^s(L_v^1) \leq r^s(L_v^0)\ \forall s \in \mathcal{RES} \tag{4.77}$$

Si on combine l'élimination de 2-cycles et les ng-routes, alors l'étiquette $L_v^0 \in \mathcal{F}_{\mathcal{GS}}(os,v) \cap \mathcal{S}_{\mathcal{GS}}$ où $pred(L_v^3) \in C(h)$ peut être éliminée si $\exists L_v^1, L_v^2 \in \mathcal{F}_{\mathcal{GS}}(os,v) \cap \mathcal{S}_{\mathcal{GS}}$ et que les conditions suivantes sont respectées.

Definition 11. *Règle de dominance agrégée avec ressources de visite et ajout de SRI - SPPRC avec voisinage et élimination de 2-cycles*

$$Client(\bar{v}(L_v^1)) = Client(\bar{v}(L_v^2)) = Client(\bar{v}(L_v^0)) \tag{4.78}$$

$$\bar{c}(L_v^1) - \sum_{g \in \mathscr{G}_{k,1,0}} \nu_k^g \leq \bar{c}(L_v^0) \text{ et } \bar{c}(L_v^2) - \sum_{g \in \mathscr{G}_{k,2,0}} \nu_k^g \leq \bar{c}(L_v^0) \tag{4.79}$$

$$r^s(L_v^1) \leq r^s(L_v^0)\ \forall s \in \mathcal{RES} \setminus \{visit_j\} \text{ et}$$
$$r^s(L_v^2) \leq r^s(L_v^0)\ \forall s \in \mathcal{RES} \setminus \{visit_i\} \tag{4.80}$$

où

$$Client(pred(L_v^0)) = h,\ Client(pred(L_v^2)) = i,\ Client(pred(L_v^1)) = j$$

Bien que l'on ne considère pas γ_p^g explicitement comme une ressource dans $\mathcal{RES}$, cette nouvelle règle rend quand même l'élimination d'étiquettes plus difficile. On doit effectivement considérer ce coefficient lors de la comparaison des coûts réduits des chemins partiels associés aux étiquettes.

Ces trois règles s'appliquent uniquement pour l'algorithme avant lorsqu'on prolonge les étiquettes de os vers os'. On doit se contenter de règles de dominance standard (non agrégée) pour les étiquettes arrières. Ceci donne lieu aux règles qui permettent d'éliminer l'étiquette L_v^0 :

Definition 12. *Règle de dominance standard avec élimination de 2-cycles sans ressources de visite et ajout de SRI*

$$\bar{v}(L_v^1) = \bar{v}(L_v^2) = \bar{v}(L_v^0) \tag{4.81}$$

$$\bar{c}(L_v^1) - \sum_{g \in \mathscr{G}_{k,1,0}} \nu_k^g \leq \bar{c}(L_v^0) \text{ et } \bar{c}(L_v^2) - \sum_{g \in \mathscr{G}_{k,2,0}} \nu_k^g \leq \bar{c}(L_v^0) \tag{4.82}$$

$$r^{dem}(L_v^1) \leq r^{dem}(L_v^0) \text{ et } r^{dem}(L_v^2) \leq r^{dem}(L_v^0) \tag{4.83}$$

$$pred(L_v^1) \neq pred(L_v^2) \neq pred(L_v^0) \tag{4.84}$$

Definition 13. *Règle de dominance standard avec ressources de visite - SPPRC avec voisinage sans élimination de 2-cycles et ESPPRC et ajout de SRI*

$$\bar{v}(L_v^1) = \bar{v}(L_v^0) \tag{4.85}$$

$$\bar{c}(L_v^1) - \sum_{g \in \mathscr{G}_{1,0}} \nu_k^g \leq \bar{c}(L_v^0) \tag{4.86}$$

$$r^s(L_v^1) \leq r^s(L_v^0)\ \forall s \in \mathcal{RES} \tag{4.87}$$

Definition 14. *Règle de dominance standard avec ressources de visite, élimination de 2-cycles et ajout de SRI - SPPRC avec voisinage et élimination de 2-cycles*

$$\bar{v}(L_v^1) = \bar{v}(L_v^2) = \bar{v}(L_v^0) \tag{4.88}$$

$$pred(L_v^1) \neq pred(L_v^2) \neq pred(L_v^0) \tag{4.89}$$

$$r^s(L_v^1) \leq r^s(L_v^0)\ \forall s \in \mathcal{RES} \setminus \{visit_j\}\ et$$

$$r^s(L_v^2) \leq r^s(L_v^0)\ \forall s \in \mathcal{RES} \setminus \{visit_i\} \tag{4.90}$$

$$\bar{c}(L_v^1) - \sum_{g \in \mathcal{G}_{k,1,0}} \nu_k^g \leq \bar{c}(L_v^0)\ et\ \bar{c}(L_v^2) - \sum_{g \in \mathcal{G}_{k,2,0}} \nu_k^g \leq \bar{c}(L_v^0) \tag{4.91}$$

où

$$Client(pred(L_v^0)) = h,\ Client(pred(L_v^2)) = i,\ Client(pred(L_v^1)) = j$$

Modifications futures pour l'algorithme tabou

Il est possible d'exploiter la nouvelle définition du coût réduit des routes selon les inégalités valides ajoutées pour améliorer l'algorithme tabou utilisé. Les figures 4.7 et 4.8 démontrent notamment que la recherche taboue doit trouver les nouveaux arcs entre chaque nœud se situant après l'insertion ou la suppression d'un client dans une route afin de déterminer si la nouvelle route produite aura un coût réduit négatif. Afin d'accélérer cette procédure, il pourrait être avantageux d'utiliser une stratégie similaire à celle de (Gendreau *et al.*, 1995) et d'utiliser des bornes rapidement calculables pour évaluer le coût d'une route. Il serait notamment possible d'utiliser les approximations suivantes :

Soit deux routes dans $\mathcal{G}$:

- $p = (i_1 = o, i_2, \ldots, i_{h-1}, i_h, i_{h+1}, \ldots i_{m_p})$
- $p' = (j_1 = o, j_2, \ldots, j_{h-1}, j_h, j_{h+1}, \ldots j_{m_{p'}})$

où p' est la route obtenue à partir de p en enlevant le client i_h. On a donc $i_s = j_s\ \forall s \in \{1, \ldots, h-1\}$ et $i_{s+1} = j_s\ \forall s \in \{h, \ldots, m_p - 1\}$ et $m_p = m_{p'} + 1$.

On se rappelle que le coût réduit d'une route p à la k^e itération de l'algorithme de génération de colonnes est donné par :

$$\bar{c}_k(p) = \sum_{\substack{(i,j) \in p \\ a \in C(i) \\ b \in C(j)}} (\hat{c}_{ab} - \pi_{jk} - \sum_{f \in \mathcal{H}} \bar{\beta}_{ij}^f \sigma_{ijk}^f) - \varphi_k - \sum_{g \in \mathcal{G}} \gamma_p^g \nu_k^g$$

où $\hat{c}_{ab} = c_{ij}$+EFC(μ_i+$E[\xi_j], j$) représente le coût total espéré de visite de l'arc $(a, b) \in \mathcal{AS}$. Le choix de cet arc indique qu'on visite le client i puis le client j dans $\mathcal{G}$ si on a accumulé une demande de μ_i au nœud i. Si on considère séparément le coût espéré d'échec et que l'on pose $\tilde{c}_{ijk} = c_{ij} - \pi_{jk} - \sum_{f \in \mathcal{H}} \bar{\beta}^f_{ij} \sigma^f_{ijk}$, alors on peut également exprimer le coût réduit de cette route par :

$$\bar{c}_k(p) = \sum_{(i,j) \in p} (\tilde{c}_{ijk} + \text{EFC}(\mu_j, j)) - \varphi_k - \sum_{g \in \mathcal{G}} \gamma^g_p \nu^g_k$$

On trouve donc (on omet l'indice k par souci de parcimonie et parce qu'on considère deux routes à la même itération de génération de colonnes) :

$$\begin{aligned}
\bar{c}(p) - \bar{c}(p') = & \sum_{(i,j) \in p} (\tilde{c}_{ij} + \text{EFC}(\mu_j, j)) - \varphi - \sum_{g \in \mathcal{G}} \gamma^g_p \nu^g \\
& - \left(\sum_{(i,j) \in p'} (\tilde{c}_{ij} + \text{EFC}(\mu_j, j)) - \varphi - \sum_{g \in \mathcal{G}} \gamma^g_{p'} \nu^g \right) && (4.92) \\
= & \tilde{c}_{i_{h-1}, i_h} + \tilde{c}_{i_h, i_{h+1}} - \tilde{c}_{i_{h-1}, i_{h+1}} \\
& + \sum_{s=h+1}^{m_p} (\text{EFC}(\mu_{i_s}, i_s) - \text{EFC}(\mu_{i_s} - E[\xi_{i_h}], i_s)) + \text{EFC}(\mu_{i_h}, i_h) \\
& + \sum_{g \in \mathcal{G}} (\gamma^g_{p'} - \gamma^g_p) \nu^g && (4.93) \\
\geq & \tilde{c}_{i_{h-1}, i_h} + \tilde{c}_{i_h, i_{h+1}} - \tilde{c}_{i_{h-1}, i_{h+1}} && (4.94)
\end{aligned}$$

En effet, EFC(μ_i, i) − EFC($\mu_i - E[\xi_h], i$) ≥ 0 $\forall i, h \in \mathcal{N}$, car le coût espéré d'échec à tout client est croissant selon la demande cumulée à ce dernier. De plus, $\gamma^g_{p'} - \gamma^g_p \leq 0$ et $\nu^g \leq 0$ $\forall g \in \mathcal{G}$, car dans notre implémentation précise, on considère uniquement les inégalités de clique et que le coefficient de la route p ne peut être inférieur à celui de la route p' ; et ce pour toute contrainte de ce type, car la route p couvre tous les clients de p' en plus de couvrir le client i_k. Par ailleurs, EFC(μ_i, i) ≥ 0 $\forall i \in \mathcal{N}$, car la probabilité d'échec à un client donné est toujours plus grande ou égale à zéro.

Il est donc possible d'utiliser :

$$\Delta^-_k = \tilde{c}_{i_{k-1}, i_k} + \tilde{c}_{i_k, i_{k+1}} - \tilde{c}_{i_{k-1}, i_{k+1}} \tag{4.95}$$

comme borne inférieure sur l'épargne obtenue par la suppression du client i_k dans la route

p. En d'autres mots, on peut ajouter la colonne correspondant au chemin p' si $\bar{c}(p) - \Delta_k^- < 0$.

On note également que :

$$\begin{aligned}
\bar{c}(p) - \bar{c}(p') &= \sum_{(i,j)\in p} (\tilde{c}_{ij} + \mathrm{EFC}(\mu_j, j)) - \varphi_k - \sum_{g\in\mathscr{G}} \gamma_p^g \nu_k^g \\
&\quad - \left(\sum_{(i,j)\in p'} (\tilde{c}_{ij} + \mathrm{EFC}(\mu_j, j)) - \varphi_k - \sum_{g\in\mathscr{G}} \gamma_{p'}^g \nu_k^g \right) \qquad (4.96) \\
&= \tilde{c}_{i_{k-1},i_k} + \tilde{c}_{i_k,i_{k+1}} - \tilde{c}_{i_{k-1},i_{k+1}} \\
&\quad + \sum_{s=k}^{m_p-1} (\mathrm{EFC}(\mu_{j_s} + E[\xi_{i_k}], j_s) - \mathrm{EFC}(\mu_{j_s}, i_s)) + \mathrm{EFC}(\mu_{j_{k-1}} + E[\xi_{i_k}], i_k) \\
&\quad + \sum_{g\in\mathscr{G}} (\gamma_{p'}^g - \gamma_p^g)\nu_k^g \qquad (4.97) \\
&\leq \tilde{c}_{i_{k-1},i_k} + \tilde{c}_{i_k,i_{k+1}} - \tilde{c}_{i_{k-1},i_{k+1}} \\
&\quad + \sum_{s=k}^{m_p-1} \mathrm{EFC}(\mu_{j_s} + E[\xi_{i_k}], j_s) + \mathrm{EFC}(\mu_{j_{k-1}} + E[\xi_{i_k}], i_k) \\
&\quad - \sum_{g\in\mathscr{G}} \nu_k^g \,^3 \qquad (4.98)
\end{aligned}$$

3. Contrairement au cas avec suppression, on exprime la borne sur la variation du coût réduit en fonction du coût espéré d'échec sur la route p' (on utilise j_s et non i_s). On utilise cette convention, car on suppose qu'on part de la route p lorsque l'on supprime le client i_k et que l'on part de p' lorsqu'on l'ajoute.

On peut donc utiliser :

$$\begin{aligned}\Delta_k^+ = \tilde{c}_{i_{k-1},ik} + \tilde{c}_{i_k,ik+1} - \tilde{c}_{i_{k-1},ik+1} \\ + \sum_{s=k}^{m_p-1} (\text{EFC}(\mu_{j_s} + E[\xi_{i_k}], j_s)) + \text{EFC}(\mu_{j_{k-1}} + E[\xi_{i_k}], i_k) \\ - \sum_{g \in \mathscr{G}} \nu_k^g \end{aligned} \tag{4.99}$$

comme borne supérieure sur l'augmentation du coût réduit si on ajoute le client i_k à la route p'. En d'autres mots, on peut ajouter la colonne correspondant au chemin p si $\bar{c}(p') + \Delta_k^+ < 0$. Ce calcul se révèle presqu'aussi complexe que le calcul du coût réduit réel de la nouvelle route. Même si on simplifie ce calcul en relaxant cette borne, ceci rendra l'identification de colonnes de coût réduit plus difficile.

4.4.4 Ajout des inégalités valides dans PM

Il existe un nombre exponentiel de CC et $O(n^3)$ de SRI. Le problème maître pourrait donc comprendre un nombre exponentiel de variables de décision et de contraintes si on les ajoutait toutes. Par conséquent, comme on le faisait pour les routes réalisables, on considère uniquement un sous-ensemble restreint des inégalités valides à chaque itération de l'algorithme de génération de colonnes. On définit $\mathscr{H}_k$ et $\mathscr{G}_k$ comme le sous-ensemble des inégalités valides pouvant être introduites directement dans $DEPs$ et dans PM respectivement identifiées à l'itération k de l'algorithme de génération de colonnes. Suite à l'ajout de ce deux types d'inégalités, on produit un problème maître restreint augmenté PMR_k à l'itération k :

$$
\begin{aligned}
& \min_{\lambda} && \sum_{p \in \mathscr{P}_k} \hat{c}_p \lambda_p \\
(PMR_k) \quad & \text{s. à.} && \sum_{p \in \mathscr{P}_k} \alpha_{ip} \lambda_p = 1 \quad \forall i \in \mathcal{N} \\
& && \sum_{p \in \mathscr{P}_k} \lambda_p \geq k(\mathcal{N}) \\
& && \sum_{p \in \mathscr{P}_k} \beta_{ij}^f \lambda_p \geq k(S^f) \quad \forall f \in \mathcal{H}_k \\
& && \sum_{p \in \mathscr{P}_k} \gamma_p^g \lambda_p \leq \gamma_0^g \quad \forall g \in \mathscr{G}_k \\
& && \lambda_p \in \{0,1\} \; \forall p \in \mathscr{P}_k
\end{aligned}
$$

4.5 Règles de branchement

Lorsque l'algorithme de séparation ne parvient plus à trouver d'inégalités valides suffisamment violées par la solution fractionnaire en cours, on procède à un branchement dichotomique. Cette étape consiste à séparer S_k^l, le domaine réalisable actuel de PMR_k^l en deux domaines de solutions disjoints.

Il existe de nombreuses décisions de branchement pouvant être prises. Ces dernières peuvent avoir un impact assez important sur le temps de résolution total. Par exemple, dans la section 5.2, on montre que certaines instances peuvent seulement être résolues en moins de 20 minutes si on utilise une règle de branchement particulière.

Comme nous nous inspirons des travaux de (Christiansen et Lysgaard, 2007), il semble raisonnable d'expérimenter avec leurs règles de branchement. Dans leur article, ces derniers proposent d'utiliser deux méthodes : l'une consiste à brancher sur la demande cumulée à un client donné alors que la deuxième branche sur une variable de flot unique (sur la valeur d'un arc donné). Ces règles et leurs problèmes associés sont expliqués ci-bas.

4.5.1 Branchement sur demande cumulée

La première méthode suggérée par (Christiansen et Lysgaard, 2007) s'inspire du branchement sur les fenêtres de temps (Gelinas *et al.*, 1995) et consiste à brancher sur la demande cumulée à un client donné. On doit trouver un client i visité plus d'une fois et imposer la contrainte $\mu_i > t$ dans un nœud de branchement et $\mu_i \leq t$ dans l'autre noeud de branchement

où $t < Q$ représente un seuil quelconque.

La décision est réalisée en enlevant tous les nœuds $\mathscr{V}(\mu, i)$ tels que $\mu > t$ dans $\mathcal{GS}$ ainsi que les arcs incidents dans le premier nœud de branchement et en enlevant les nœuds $\mathscr{V}(\mu, i)$ tels que $\mu \leq t$ dans $\mathcal{GS}$ ainsi que leur arcs incidents dans le second nœud de branchement (voir la figure 4.10).

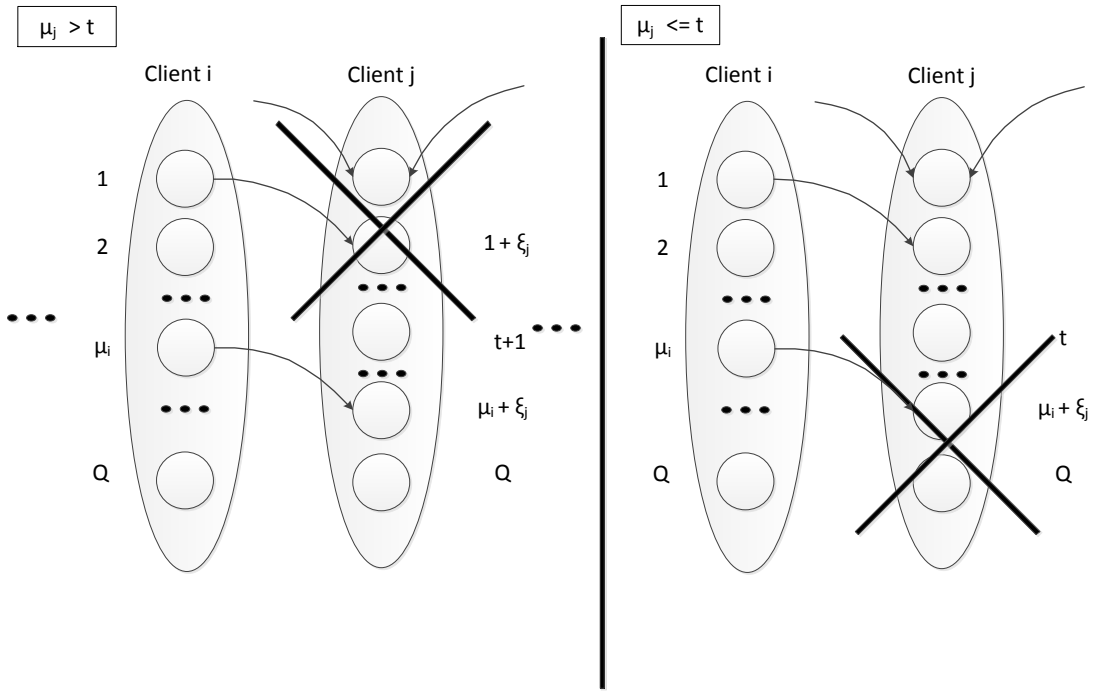

Figure 4.10 Branchement dichotomique basé sur la demande cumulée ($\mu_i \leq t$ et $\mu_i > t$)

On choisit le seuil t comme la demande cumulée permettant d'obtenir une division du flot la plus près « égale » possible. En d'autres mots, si on considère le client i, on choisit t de manière à minimiser $\{|\sum_{(a,b)\in\delta^-(C(i))} \hat{x}_{ab} - 0.5| : v = \mathscr{V}(\mu_i, i), \mu_i \leq t, \alpha_{abp}\lambda_p > 0\}$. Afin de s'assurer de rendre la solution fractionnaire actuelle λ_k^l irréalisable dans les deux domaines, on choisit un client i visité plus d'une fois à différentes demandes cumulées. On s'assure ainsi que la règle $\mu_i \leq t$ interdira la solution en cours. Ceci ne serait pas nécessairement le cas si deux chemins différents p et p' passaient tous deux par i avec la même demande cumulée ($\mu_i^p = \mu_i^{p'}$).

4.5.2 Branchement sur arcs

La seconde méthode utilisé par (Christiansen et Lysgaard, 2007) consiste simplement à identifier un arc $(a, b) \in \mathcal{AS}^l$ visité par un chemin dont la valeur de la variable de décision dans PM est fractionnaire et d'enlever cet arc de $\mathcal{GS}$ dans le premier nœud de branchement. On impose aussi la présence de cet arc dans $\mathcal{GS}$ dans le second nœud de branchement en enlevant tous les arcs (k, j) tels que $k \neq i$ dans $\mathcal{GS}$.

4.5.3 Incompatibilité des règles de branchement avec la dominance agrégée

Bien qu'intéressantes, les deux règles de branchement 4.5.1 et 4.5.2 sont incompatibles avec la règle de dominance agrégée présentée dans la section sur l'algorithme de plus court chemin 4.2.13. En effet, avec ces règles de branchement, la règle agrégée n'est même pas applicable pour l'algorithme avant, car il est impossible de garantir que pour deux étiquettes L_u^1 et L_v^2 telles que $\mu_1 < \mu_2, r^s(L_u^1) \leq r^s(L_v^2) \forall s \in \mathcal{RES}$ et $\bar{v}(L_u^1) = \bar{v}(L_v^2)$ et $\bar{c}(L_u^1) \leq \bar{c}(L_v^2)$ toute extension réalisable de L_u^2 sera dominée par toute extension réalisable de L_v^1.

Cette situation est bien illustrée dans la figure 4.11. On observe notamment que $\mathcal{E}(L_v^2) \not\subseteq \mathcal{E}(L_u^1)$ (où des extensions représentent un ensemble de clients), car l'étiquette L_v^2 peut visiter le client j alors que L_u^1 ne peut pas. Il est donc possible que l'étiquette dominée mène à une étiquette de la source au puits de coût réduit inférieur à celui de toutes les étiquettes issues des extensions de l'étiquette dominante. On ne peut donc pas éliminer l'étiquette L_v^2 comme on le ferait si on appliquait une des trois règles de dominance agrégée. On peut appliquer un raisonnement similaire avec le branchement basé sur les arcs.

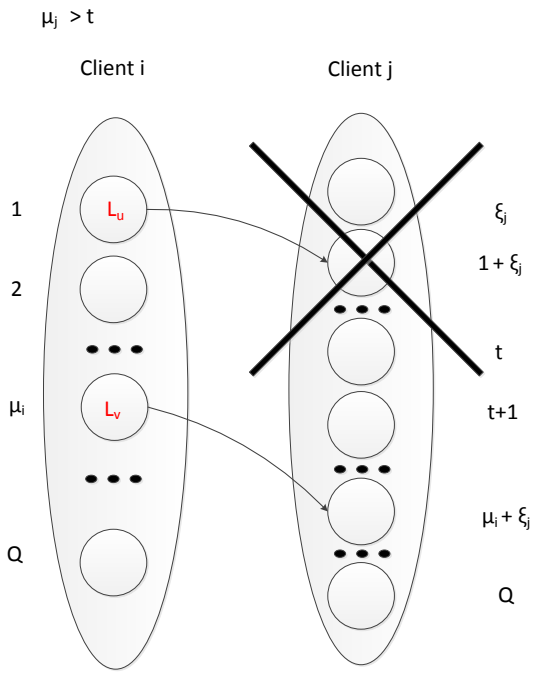

Figure 4.11 Incompatibilité du branchement sur la demande cumulative et de la dominance agrégée

Sommairement, les règles de branchement utilisées conjointement avec les règles de dominance agrégée devraient agir sur des clients entiers (donc sur l'ensemble des arcs entrant et sortant dans un ensemble $C(i)$ pour tout client $i \in \mathcal{N}$) plutôt que sur des arcs individuels.

4.5.4 Branchement sur les inter-tâches

Afin de contourner ces difficultés, nous adoptons une méthode de branchement basée sur les inter-tâches (séquence de deux tâches consécutives). On considère une route $p_v = (o = i_1, i_2, ..., i_{|p_v|})$ où $i_d \in \mathcal{N};\ d = 1...|p_v|$ empruntée par un véhicule v dans $\tilde{\hat{x}}$, la solution fractionnaire de $DEPs$. On pose également $\mathcal{AS}^l_v(i_k \rightarrow i_{k+1}) = \{(a,b) \in \mathcal{AS}^l : a \in C(i_k) \wedge b \in C(i_{k+1}) \wedge (i_k, i_{k+1}) \in p_v; 1 \leq k \leq |p_v| - 1\}$ comme l'ensemble des arcs de $\mathcal{GS}$ choisis par la route et $\{(i_k, i_{k+1}) \in p_v; 1 \leq k \leq |p_v| - 1\}$ comme l'ensemble des inter-tâches de la route.

La méthode de branchement basée sur les inter-tâches consiste à identifier une paire

de clients (i,j) successifs tels que $0 < \sum_{v\in\mathcal{V}} \sum_{(a,b)\in\mathcal{AS}_v^l(i\to j)} \tilde{x}_{ab}^v < 1$. On impose ensuite dans le premier nœud de branchement que le client j suive le client i alors qu'on interdit cette séquence dans le deuxième nœud. Cette règle est implantée en enlevant tous les arcs $(\mathscr{V}(\mu_i,i),\mathscr{V}(\mu_i+E[\xi_k],k))\ \forall k \in \mathcal{N}\setminus\{j\},\ \forall \mu_i \in \{1,2,...,Q-E[\xi_k]\}$ de $\mathcal{GS}$ dans le premier nœud de branchement et en enlevant les arcs $(\mathscr{V}(\mu_i,i),\mathscr{V}(\mu_i+E[\xi_j],j))\ \forall \mu_i \in \{1,2,...,Q-E[\xi_j]\}$ dans le second nœud de branchement.

Comme il peut exister plusieurs inter-tâches candidates, on évalue la qualité de chacune d'entre elle grâce au score $2\times\min\{\sum_{v\in\mathcal{V}} \sum_{(a,b)\in\mathcal{AS}_v^l(i\to j)} \tilde{x}_{ab}^v, 1-\sum_{v\in\mathcal{V}} \sum_{(a,b)\in\mathcal{AS}_v^l(i\to j)} \tilde{x}_{ab}^v\}$. Chaque inter-tâche a donc un score entre 0 et 1 et le score maximal de 1 est atteint lorsque le flot correspondant de l'inter-tâche est 0.5.

4.5.5 Branchement sur le flot adjacent d'un ensemble de clients

Nous avons également effectué des tests avec une autre méthode de branchement utilisée notamment par (Fukasawa *et al.*, 2006). Étant donné une solution fractionnaire $\tilde{x}$ de DEP, cette règle consiste à identifier un ensemble de clients $S \subseteq \mathcal{N}$ tel que :

$$2 < \sum_{v\in\mathcal{V}} \sum_{(i,j)\in\delta(S)} \tilde{x}_{ij}^v < 4 \tag{4.100}$$

où $\delta(S) = \delta^-(S) \cup \delta^+(S)$ représente l'ensemble des arcs dans le graphe $\mathcal{G}$ original ayant exactement un nœud dans l'ensemble $S \subseteq \mathcal{N}$.

On crée ensuite deux nœuds de branchement. Dans le premier noeud de branchement, on impose $\sum_{p\in\mathscr{P}} \bar{\alpha}_{Sp}\lambda_p \geq 4$ dans le problème maître et dans le second on impose $\sum_{p\in\mathscr{P}} \bar{\alpha}_{Sp}\lambda_p = 2$. On définit $\mathcal{AS}_p \subseteq \mathcal{AS}$ comme les arcs de $\mathcal{GS}$ couverts par la route p et $\bar{\alpha}_{Sp} = |\{(\mathscr{V}(\mu_i,i),\mathscr{V}(\mu_i+E[\xi_j],j)) \in \mathcal{AS}_p : (i,j)\in\delta(S)\}|$ comme le nombre d'arcs de la route p ayant exactement un nœud dans S (lorsqu'on considère le graphe $\mathcal{G}$ original).

On sait que ces règles de branchement sont valides, car $\sum_{(i,j)\in\delta(i)} \sum_{v\in\mathcal{V}} x_{ij}^v = 2 \quad \forall i \in \mathcal{N}$ et $\sum_{(i,j)\in\delta(S)} \sum_{v\in\mathcal{V}} x_{ij}^v = 2k(S) \quad \forall S \subseteq \mathcal{N}$ où $k(S)$ représente le nombre minimal de véhicules nécessaire pour couvrir l'ensemble S (Fukasawa *et al.*, 2006). Ainsi, le nombre d'arcs ayant exactement un nœud dans tout sous-ensemble de clients est pair et il doit exister au moins 2 arcs ayant exactement un nœud dans le sous-ensemble.

Comme toute route dans une solution entière peut couvrir chaque client $i \in S$ au plus 1

fois, chaque route peut avoir au plus 2 arcs ayant exactement un nœud dans tout ensemble de clients S. Ainsi, cette règle de branchement impose que l'ensemble S soit couvert par deux routes ou plus ou par une seule route. On considère uniquement les ensembles ayant un flot adjacent strictement plus grand que 2 et strictement plus petit que 4. Par conséquent, on s'assure de rendre la solution fractionnaire actuelle irréalisable dans les deux nœuds de branchement. On note que ces règles de branchement restent les mêmes, peu importe la définition de $\mathscr{P}$.

Tout comme pour les inter-tâches, il peut exister plusieurs candidats. On construit chaque sous-ensembles S à l'aide d'une heuristique gloutonne qui ajoute des noeuds à un ensemble pour obtenir un flot près de 3 qui ne dépasse pas 4 (voir (Lysgaard *et al.*, 2004)). Pour chaque sous-ensembles S candidat, on attribue ensuite le score $1 - |\sum_{(i,j)\in\delta(S)} \sum_{v\in\mathcal{V}} x_{ij}^v - 3|$. On attribue ainsi un score compris entre 0 et 1 qui est plus plus élevé pour les ensembles dont le flot est très différent de 3. Ceci rendra la solution actuelle irréalisable dans les deux nœuds de branchement fils et est susceptible de créer un nœud dont la borne inférieure sera similaire à la borne inférieure actuelle et un autre avec une borne inférieure plus grande. Ces ensembles sont identifiés à l'aide du module CVRPSEP (Lysgaard., 2004).

CHAPITRE 5

RÉSULTATS NUMÉRIQUES

Afin de bien comparer l'efficacité des différents algorithmes utilisés ainsi que celui de (Christiansen et Lysgaard, 2007), il est essentiel de faire des expériences numériques. Ces tests sont utiles pour évaluer l'importance réelle de plusieurs éléments comme l'algorithme d'étiquetage bidirectionnel et l'utilisation de l'algorithme tabou. Les résultats obtenus permettent également d'étudier l'interaction entre les différentes techniques et de mieux comprendre les difficultés particulières de chaque instance. Cette analyse peut éventuellement permettre d'identifier de nouvelles pistes de recherche futures pour le VRPSD et les problèmes de tournées de véhicules en général.

5.1 Paramètres Généraux

Nous utilisons les mêmes instances tirées de la littérature que celles utilisées par (Christiansen et Lysgaard, 2007). Ces problèmes proviennent notamment des instances A et P d'Augerat et al. et les instances E de Christophides et Eilon. Toutes ces instances peuvent être obtenues sur le site : http ://branchandcut.org.

Tous les tests ont été effectués sur un ordinateur avec un processeur Intel i7-2600 avec 3,4 GHz et 16 GB de RAM à l'aide du logiciel GENCOL développé au GERAD et écrit en C et en C++. Les résultats présentés dans (Christiansen et Lysgaard, 2007) ont quand à eux été obtenus à l'aide d'un processeur Pentium Centrino 1,5GHz avec 480 MB de RAM.

Afin de comparer adéquatement l'efficacité de ces deux méthodes et les temps de calcul présentés, il est utile d'utiliser un facteur de correction pour tenir compte de l'augmentation constante de la puissance des processeurs des ordinateurs au cours des années. Nous calculons ce facteur grâce aux données présentées dans (Dongarra, 2011). Ce document technique liste la quantité de millions d'opérations sur les nombres flottants par seconde (MFLOP/s) selon différents types de serveurs ayant des processeurs différents. Cette conversion suppose que les temps de calcul pour les processeurs sont approximativement linéairement proportionnels aux MFLOP/s. Par ailleurs, ces tests ont été effectués dans le cadre de résolution de systèmes d'équations denses. Bien que (Dongarra, 2011) ne présente pas des données explicites pour

les processeurs spécifiquement utilisés dans nos travaux ainsi que ceux de (Christiansen et Lysgaard, 2007), on suppose que la plupart des processeurs sont approximativement aussi rapides que les autres de la même famille avec des caractéristiques semblables. On suppose donc que MFLOP/s=796 pour (Christiansen et Lysgaard, 2007) et de MFLOP/s=1573 pour nos résultats. On a donc un ratio de 1,98. Bien qu'imparfaite, cette conversion est tout de même utile à titre indicatif. Tous les temps de calcul de (Christiansen et Lysgaard, 2007) sont divisés par ce facteur.

Comme dans (Christiansen et Lysgaard, 2007), nous divisons la demande de tous les clients ainsi que la capacité par le plus grand commun diviseur. Comme ces auteurs, nous calculons également le coût de déplacement d'un noeud à un autre comme la distance euclidienne entre ces deux points. Ces distances sont ensuite arrondies à l'entier le plus près. Le calcul du coût espéré d'échec est effectué après avoir calculé et arrondi toutes les distances.

Tous les résultats suivants ont été obtenus en utilisant les paramètres suivants :

◇ Général
- Temps maximal permis de 1200 secondes (20 minutes)

◇ Branchement/Algorithmes de séparation
- Librairie CVRPSEP utilisée pour identifier les inégalités valides violées par la solution fractionnaire en cours (Lysgaard., 2004). Tous les algorithmes de séparation sont exécutés sur le graphe $\mathcal{G}$ original.
- Inégalités valides ajoutées : CC et SRI ;
- Seulement 1 type d'inégalité valide ajouté à chaque fois qu'on parvient à en identifier ;
  - On tente d'abord d'identifier des coupes de capacité
  - Si l'algorithme échoue à en identifier, il tente ensuite d'identifier des SRI
- Aucune limite sur le nombre total de CC ou de SRI générées au cours de l'algorithme entier ;
- Ajout de toutes les inégalités CC identifiées (seuil de violation minimale de 0) à un nœud de branchement ;
- Ajout des SRI identifiées qui ont une violation supérieure à 0,1 à tout nœud de branchement ;
- Aucune limite sur le nombre total de CC générées à chaque fois qu'on parvient à en identifier ;
- Maximum de 15 SRI ajoutées à chaque noeud de branchement ;

- Priorité accordée à l'identification d'inégalités valides violées par rapport aux décisions de branchement ;
- Branchement dichotomique choisi si la violation maximale des coupes identifiées est inférieure ou égal au seuil critique de violation ;
- Deux règles de branchement sont considérées et choisies selon les critères décrits à la section 4.5.2 :
  - Branchement sur les inter-tâches ;
  - Branchement sur le flot adjacent d'un ensemble de clients.
- Sélection du nœud de branchement dont la valeur de la plus borne inférieure du nœud père est minimale (« best first »).

◇ Algorithme de recherche taboue

- Nombre maximal de colonnes pouvant être générées par l'algorithme à chaque itération de l'algorithme de génération de colonnes : 500 ;
- Taille de la liste taboue : 10.

5.2 Résultats des différents algorithmes

Comme notre algorithme se base sur plusieurs techniques à la fin pointe de l'état de l'art en matière de résolution de problèmes de tournées de véhicule, nous avons effectué plusieurs tests afin de calibrer le modèle et identifier les composantes utiles et superflues. Nous nous sommes particulièrement penchés sur le type de sous-problème (élémentaire ou ng-routes), l'utilisation d'un algorithme tabou et l'utilisation de l'algorithme d'étiquetage bidirectionnel. Les résultats numériques sont groupés selon la taille des instances et sont présentés dans les tableaux 5.1 5.2 et 5.3. Les colonne de ces tableaux sont données par :

- **Instance** : Le nom de l'instance.
- **UB** : Une borne supérieure sur la valeur optimale de l'instance. Le signe * indique que cette borne est optimale.
- **Temps** : le temps total nécessaire pour résoudre le problème (en secondes). Le signe # indique que l'on n'est pas parvenu à résoudre l'instance en moins de 20 minutes. Les valeurs indiquées en **gras** indiquent les temps minimaux.
- **Élémentaire** : résultats obtenus avec le problème de plus court chemin élémentaire (ESPPRC).

- ng**-route (10)** : résultats obtenus avec le problème de plus court chemin avec ng-routes et $\Delta(N_i) = 10 \; \forall i \in N$.
- **Minimum** et **Maximum** : temps minimal et maximal respectivement pour la résolution de l'instance avec le sous-problème spécifié.
- **Lysgaard** : Meilleur temps et borne supérieure publiés dans (Christiansen et Lysgaard, 2007).
- $T, T_\emptyset$: indique que l'algorithme de recherche tabou est utilisé ou non.
- $B, B_\emptyset$: indique que l'algorithme de plus court chemin bidirectionnel est utilisé ou non.

Tableau 5.1 Résultats finaux pour les petites instances (jusqu'à 33 clients)

Instance		Élémentaire	ng-route (10)				Minimum	Maximum	Lysgaard
		TB	$\bar{T}_{\emptyset}B_{\emptyset}$	$T_{\emptyset}B$	$T\bar{B}_{\emptyset}$	TB			
A-n32-k5	Time :	**12,1**	56,6	20,8	34,0	24,1	12,1	56,6	142,4
	UB :	853,6*	853,6*	853,6*	853,6*	853,6*			853,6*
A-n33-k5	Temps :	6,0	6,4	10,2	**4,0**	5,3	4,0	10,2	4,0
	UB :	704,2*	704,2*	704,2*	704,2*	704,2*			704,2*
A-n33-k6	Temps :	5,0	4,2	4,2	**3,8**	4,9	3,8	5,0	24,7
	UB :	793,9*	793,9*	793,9*	793,9*	793,9*			793,9*
E-n22-k4	Temps :	0,2	**0,1**	**0,1**	**0,1**	**0,1**	0,1	0,2	0,5
	UB :	411,5*	411,5*	411,5*	411,5*	411,5*			411,6*
E-n33-k4	Temps :	34,2	**19,7**	28,2	21,1	32,6	19,7	34,2	43,4
	UB :	850,2*	850,2*	850,2*	850,2*	850,2*			850,3*
P-n16-k8	Temps :	**0,0**	**0,0**	**0,0**	**0,0**	**0,0**	0,0	0,0	0,0
	UB :	512,8*	512,8*	512,8*	512,8*	512,8*			512,8*
P-n19-k2	Temps :	6,8	**6,5**	7,2	19,3	9,7	6,5	19,3	77,3
	UB :	224,0*	224,0*	224,0*	224,0*	224,0*			224,1*
P-n20-k2	Temps :	**90,7**	#	174,8	#	128,8	90,7	#	177,8
	UB :	233,0*	233,0	233,0*	233,0	233,0*			233,1*
P-n21-k2	Temps :	1,4	1,7	2,1	**1,2**	2,2	1,2	2,2	2,5
	UB :	218,9*	218,9*	218,9*	218,9*	218,9*			219,0*
P-n22-k2	Temps :	**45,6**	241,6	84,8	180,6	46,7	45,6	241,6	110,6
	UB :	231,2*	231,2*	231,2*	231,2*	231,2*			231,3*
P-n22-k8	Temps :	**0,0**	**0,0**	**0,0**	**0,0**	**0,0**	0,0	0,0	0,0
	UB :	681,0*	681,0*	681,0*	681,0*	681,0*			681,1*
P-n23-k8	Temps :	**0,0**	**0,0**	**0,0**	**0,0**	**0,0**	0,0	0,0	0,5
	UB :	619,5*	619,5*	619,5*	619,5*	619,5*			619,5*

Tableau 5.2 Résultats finaux pour les instances de taille moyenne (34 à 48 clients)

Instance		Élémentaire	*ng*-route (10)				Minimum	Maximum	Lysgaard
		TB	$T_{\emptyset}B_{\emptyset}$	$T_{\emptyset}B$	$TB_{\emptyset}$	TB			
A-n34-k5	Temps :	9,2	9,3	8,0	**7,9**	9,0	7,9	9,3	#
	UB :	826,8*	826,8*	826,8*	826,8*	826,8*			827,9
A-n36-k5	Temps :	255,6	212,2	**40,3**	148,0	46,9	40,3	255,6	#
	UB :	858,7*	858,7*	858,7*	858,7*	858,7*			907,5
A-n37-k5	Temps :	35,2	31,8	28,5	**18,5**	23,4	18,5	35,2	#
	UB :	708,3*	708,3*	708,3*	708,3*	708,3*			708,3
A-n37-k6	Temps :	52,2	**18,1**	24,2	19,0	22,8	18,1	52,2	#
	UB :	1030,7*	1030,7*	1030,7*	1030,7*	1030,7*			1030,8
A-n38-k5	Temps :	76,4	66,8	55,5	43,5	**42,8**	42,8	76,4	#
	UB :	775,1*	775,1*	775,1*	775,1*	775,1*			778,1
A-n39-k5	Temps :	5,1	4,6	**3,9**	5,2	5,0	3,9	5,2	1,5
	UB :	869,1*	869,1*	869,1*	869,1*	869,1*			869,2*
A-n39-k6	Temps :	41,7	**12,0**	17,4	17,0	19,3	12,0	41,7	140,9
	UB :	876,6*	876,6*	876,6*	876,6*	876,6*			876,6*
A-n44-k6	Temps :	533,6	253,8	125,9	171,3	**125,0**	125,0	533,6	#
	UB :	1025,4*	1025,4*	1025,4*	1025,4*	1025,4*			1025,5
A-n45-k6	Temps :	205,9	247,6	**81,3**	191,6	86,3	81,3	247,6	#
	UB :	1026,7*	1026,7*	1026,7*	1026,7*	1026,7*			1096,2
A-n45-k7	Temps :	38,2	**23,2**	25,7	25,5	30,5	23,2	38,2	445,5
	UB :	1264,8*	1264,8*	1264,8*	1264,8*	1264,8*			1264,8*
A-n46-k7	Temps :	24,8	24,1	20,5	26,9	**18,7**	18,7	26,9	#
	UB :	1002,2*	1002,2*	1002,2*	1002,2*	1002,2*			1002,4
A-n48-k7	Temps :	37,9	31,5	**27,5**	33,3	**27,5**	27,5	37,9	#
	UB :	1187,1*	1187,1*	1187,1*	1187,1*	1187,1*			1248,3
P-n40-k5	Temps :	11,9	**4,3**	5,7	5,7	6,9	4,3	11,9	4,5
	UB :	472,5*	472,5*	472,5*	472,5*	472,5*			472,5*
P-n45-k5	Temps :	**1021,2**	#	#	#	1193,8	1021,2	#	#
	UB :	533,5*	1271,0	534,4	1090,4	533,5*			546,0

Tableau 5.3 Résultats finaux pour les grandes instances (50 clients et plus)

Instance		Élémentaire	ng-route (10)				Minimum	Maximum	Lysgaard
		TB	$T_\emptyset B_\emptyset$	$T_\emptyset B$	$TB_\emptyset$	TB			
A-n53-k7	Temps :	#	505,7	311,9	**295,1**	512,4	295,1	#	#
	UB :	1124,2	1124,2*	1124,2*	1124,2*	1124,2*			1180,1
A-n54-k7	Temps :	#	350,6	305,6	**298,3**	310,5	298,3	#	#
	UB :	3859,0	1287,0*	1287,0*	1287,0*	1287,0*			1342,9
A-n55-k9	Temps :	55,4	**24,6**	32,4	29,8	33,2	24,6	55,4	#
	UB :	1179,1*	1179,1*	1179,1*	1179,1*	1179,1*			1264,2
A-n60-k9	Temps :	#	#	#	#	#	#	#	#
	UB :	3716,0	3695,0	3711,0	3705,0	3565,0			1608,4
E-n51-k5	Temps :	#	#	#	#	#	#	#	#
	UB :	1761,0	1627,0	1627,0	567,9	567,9			553,3
P-n50-k10	Temps :	53,1	**32,2**	42,7	32,9	54,6	32,2	54,6	#
	UB :	758,7*	758,7*	758,7*	758,7*	758,7*			792,2
P-n50-k7	Temps :	59,2	22,7	33,9	**22,4**	40,4	22,4	59,2	#
	UB :	582,3*	582,3*	582,3*	582,3*	582,3*			606,4
P-n50-k8	Temps :	79,4	**20,7**	34,3	26,9	35,7	20,7	79,4	#
	UB :	669,2*	669,2*	669,2*	669,2*	669,2*			724,7
P-n51-k10	Temps :	13,8	**6,0**	7,9	7,7	10,0	6,0	13,8	217,2
	UB :	809,7*	809,7*	809,7*	809,7*	809,7*			809,7*
P-n55-k10	Temps :	30,0	**13,2**	20,9	16,0	26,4	13,2	30,0	#
	UB :	742,4*	742,4*	742,4*	742,4*	742,4*			797,2
P-n55-k15	Temps :	26,1	**10,7**	17,8	13,3	19,9	10,7	26,1	400,0
	UB :	1068,0*	1068,0*	1068,0*	1068,0*	1068,0*			1068,0*
P-n55-k7	Temps :	173,0	92,9	111,2	**91,5**	103,8	91,5	173,0	#
	UB :	588,5*	588,5*	588,5*	588,5*	588,5*			616,4
P-n60-k10	Temps :	926,7	459,8	505,9	**275,9**	417,5	275,9	926,7	#
	UB :	803,6*	803,6*	803,6*	803,6*	803,6*			831,2
P-n60-k15	Temps :	10,5	**3,2**	4,8	4,6	7,0	3,2	10,5	#
	UB :	1085,4*	1085,4*	1085,4*	1085,4*	1085,4*			1085,5

5.2.1 Correction des instances

La division des demandes et de la capacité par le plus grand commun diviseur suggérée par (Christiansen et Lysgaard, 2007) ne devrait pas être utilisée pour le VRPSD, car elle produit une solution qui ne concorde pas avec le modèle initial. Bien que cette procédure ne change pas la structure des instances dans le cas déterministe, elle a une influence importante sur le calcul de la probabilité d'échec à chaque noeud et change donc le coût total espéré des arcs dans le graphe espace-état $\mathcal{GS}$.

La division des demandes change donc non seulement le coût de la solution optimale, mais également la nature de celle-ci. La solution varie considérablement lorsque le coût total d'échec est élevé par rapport au coût de déplacement initial. Cette observation est attribuable au fait que la plupart des routes de la solution optimale (sans division) ont un coût espéré d'échec plus petit que les routes correspondantes dans le problème stochastique avec demandes divisées. En effet, la grande majorité des routes choisies dans la solution optimale des instances E-n22-k4 et P-n22-k8 ont un coût d'échec nul (leur coût est donc très près de l'entier le plus proche).

Il est vrai que le fait de ne pas diviser les demandes augmente le nombre total de noeuds dans le graphe $\mathcal{GS}$. Toutefois, un très grand nombre de noeuds ne sont pas atteignables. Ces derniers ne sont jamais visités lors de l'exécution de l'algorithme d'étiquetage et on peut les supprimer à l'aide de l'algorithme pour la résolution du problème de « subset-sum » décrit dans la section 3.3.2. Des expériences préliminaires ont démontré que sans l'élimination des noeuds superflus, il était impossible d'exécuter notre algorithme correctement à cause d'un manque de mémoire. Toutefois, après la suppression des noeuds inutiles, les réseaux contiennent exactement le même nombre de noeuds que le réseau correspondant avec la division. Les différences en termes de temps de résolution que l'on observe dans le tableau 5.4 ne peuvent donc pas être attribuées à la différence de taille. Cette différence est plutôt due au fait que les coûts d'échec réels changent considérablement la nature de l'instance et que plus de routes doivent être évaluées avant d'obtenir la solution optimale.

Tableau 5.4 Résultats correct pour les instances fausses

Instance		Élémentaire	*ng*-route (10)				Minimum	Maximum	Lysgaard
		TB	$T_\emptyset B_\emptyset$	$T_\emptyset B$	$TB_\emptyset$	TB			
P-n22-k8	Temps :	**0,0**	**0,0**	**0,0**	**0,0**	**0,0**	0,0	0,0	0,0
	UB :	592,97	592,97	592,97	592,97	592,97			681,1
E-n22-k4	Temps :	0,3	**0,2**	**0,2**	**0,2**	**0,2**	0,3	0,2	0,25
	UB :	377,10	377,10	377,10	377,10	377,10			411,6
E-n33-k4	Temps :	550,3	134,8	157,9	**129,7**	233,3	129,7	550,3	21,17
	UB :	842,00	842,00	842,00	842,00	842,00			850,3

5.2.2 Comparaison des différents algorithmes

La méthode la plus robuste et efficace est celle qui utilise les *ng*-routes avec un voisinage de 10 clients, l'algorithme tabou et l'algorithme d'étiquetage bidirectionnel - *ng*-route (10), TB. Bien qu'elle ne se révèle pas la plus rapide pour l'ensemble des instances, elle permet d'obtenir de bonnes solutions en moyenne et a un comportement assez stable d'une instance à l'autre. On note immédiatement qu'il s'agit de la seule méthode permettant de résoudre 38 instances, car les autres jeux de paramètres ne parviennent par à fermer l'instance P-n45-k5 à l'intérieur des 1200 secondes allouées (voir le tableau 5.2).

Toutefois, on ne peut se limiter à utiliser ce critère pour comparer les différentes méthodes. Pour bien comparer les différents jeux de paramètres utilisés pour l'ensemble des instances, il est utile d'utiliser des concepts introduits par (Dolan et More, 2002) pour la comparaison de logiciels d'optimisation. Cette méthodologie permet de comparer visuellement et rapidement différents algorithmes à l'aide de plusieurs facteurs comme la pire et la meilleure performance ainsi que le pourcentage d'instances résolues au total.

Afin d'utiliser ce cadre méthodologique, il est nécessaire d'introduire la notation suivante. On pose $\mathcal{AL}$ comme l'ensemble des jeux de paramètres (algorithmes) utilisés et $\mathcal{I}$ comme l'ensemble des instances (les 40 instances présentées dans (Christiansen et Lysgaard, 2007)). Pour toute paire $(algo, inst)$: $algo \in \mathcal{AL}, inst \in \mathcal{I}$, la métrique $t_{algo,ins}$ représente le temps de résolution de l'instance $inst$ par le jeu de paramètre $algo$. $t^*_{inst} = \min_{algo \in \mathcal{AL}}\{t_{algo,inst}\}$ représente le meilleur temps en secondes pour un ensemble de jeux de paramètres. On définit également ρ comme un facteur réel. On peut ensuite introduire la fonction $\omega(t, t^*, \rho)$, définie pour tout $t_{algo,inst} \geq 0, t^*_{inst} \geq 0$ et $\rho \geq 1$. Plus spécifiquement, on a :

$$\omega(t_{algo,inst}, t^*_{inst}, \rho) = \begin{cases} 1 & \text{si } t_{algo,inst} \leq \rho t^*_{inst} \\ 0 & \text{sinon} \end{cases} \tag{5.1}$$

On note que $\omega(t_{algo,inst}, t^*_{inst}, \rho) = 0$ si $t_{algo,inst} = \#$ (si l'instance n'a pas été résolue à l'intérieur de la limite de temps permise).

On définit également $\pi_{algo}(\rho)$ comme le « profil de performance » du jeux de paramètres $algo$ avec le facteur ρ. Cette métrique est obtenue à l'aide de :

$$\pi_{algo}(\rho) = \frac{\sum_{inst \in \mathcal{I}} \omega(t_{algo,inst}, t^*_{inst}, \rho)}{|\mathcal{I}|} \tag{5.2}$$

On doit calculer $\pi_{algo}(\rho)$ pour tous les algorithmes et pour ρ suffisamment grand ($\rho_{max} \leq \max_{inst \in \mathcal{I}} \{\frac{t^{min}_{inst}}{t^*_{inst}}\}$ où t^{min}_{inst} indique le pire temps de résolution pour $inst$ et ρ_{max} indique le plus grand facteur à utiliser). On trace ensuite la courbe obtenue pour chaque algorithme. Les résultats sont exposés dans la figure 5.1.

$\pi_{algo}(1)$ indique la fraction des instances pour lesquelles le jeu de paramètres $algo$ s'est révélé le plus rapide alors que $\pi_{algo}(2)$ indique la fraction des instances pour lesquelles le temps de résolution du jeu de paramètres $algo$ s'est révélé plus petit ou égal à 2 fois le meilleur temps. L'algorithme avec $\pi_{algo}(1)$ maximal indique l'algorithme qui permet de résoudre le plus d'instances le plus rapidement. Dans notre cas, cet algorithme est $algo = ng\text{-route}(10)T_\emptyset B_\emptyset$ (il permet de résoudre approximativement 43 % des instances le plus rapidement possible). On constate que $\sum_{algo \in \mathcal{AL}} \pi_{algo}(1) > 1$. On en conclut que le meilleur temps de résolution pour plusieurs instances est le même pour plusieurs algorithmes. En fait, les instances P-n16-k8, P-n22-k8 et P-n23-k8 ont toutes été résolues avec le même temps, peu importe les paramètres alors que pour l'instance E-n22-k4, seul l'algorithme élémentaire a pris plus de temps que les autres.

$\pi^{max}_{algo} = \lim_{\rho \to \infty} \pi_{algo}(\rho)$ indique le nombre maximal d'instances qui peuvent être résolues par l'algorithme $algo$. Le plus ce ratio est élevé et près de 1, le plus l'algorithme permet de résoudre d'instances à l'intérieur des 1200 secondes allouées. Comme $\pi^{max}_{ng\text{-route}(10)TB} = \max_{algo \in \mathcal{AL}} \{\pi^{max}_{algo}\}$, on en conclut que la méthode $ng\text{-route}(10)TB$ est celle qui permet de résoudre le plus d'instance.

Une courbe qui se stabilise définitivement (qui atteint π_{algo}^{max}) plus rapidement indique que $algo$ a une performance relativement stable par rapport aux autres algorithmes. Par exemple, on note que pour $algo = ng$-route(10)TB tous les problèmes résolus par cet algorithme le sont en moins de 2,3 fois le meilleur temps de résolution. Par contre, pour $algo = ng$-route(10)$T_{\emptyset}B_{\emptyset}$ toutes les instances résolues par cet algorithme le sont en moins de 5,7 fois le meilleur temps. La pire performance de l'algorithme $algo = ng$-route(10)TB est donc meilleure que celle de l'algorithme $algo = ng$-route(10)TB.

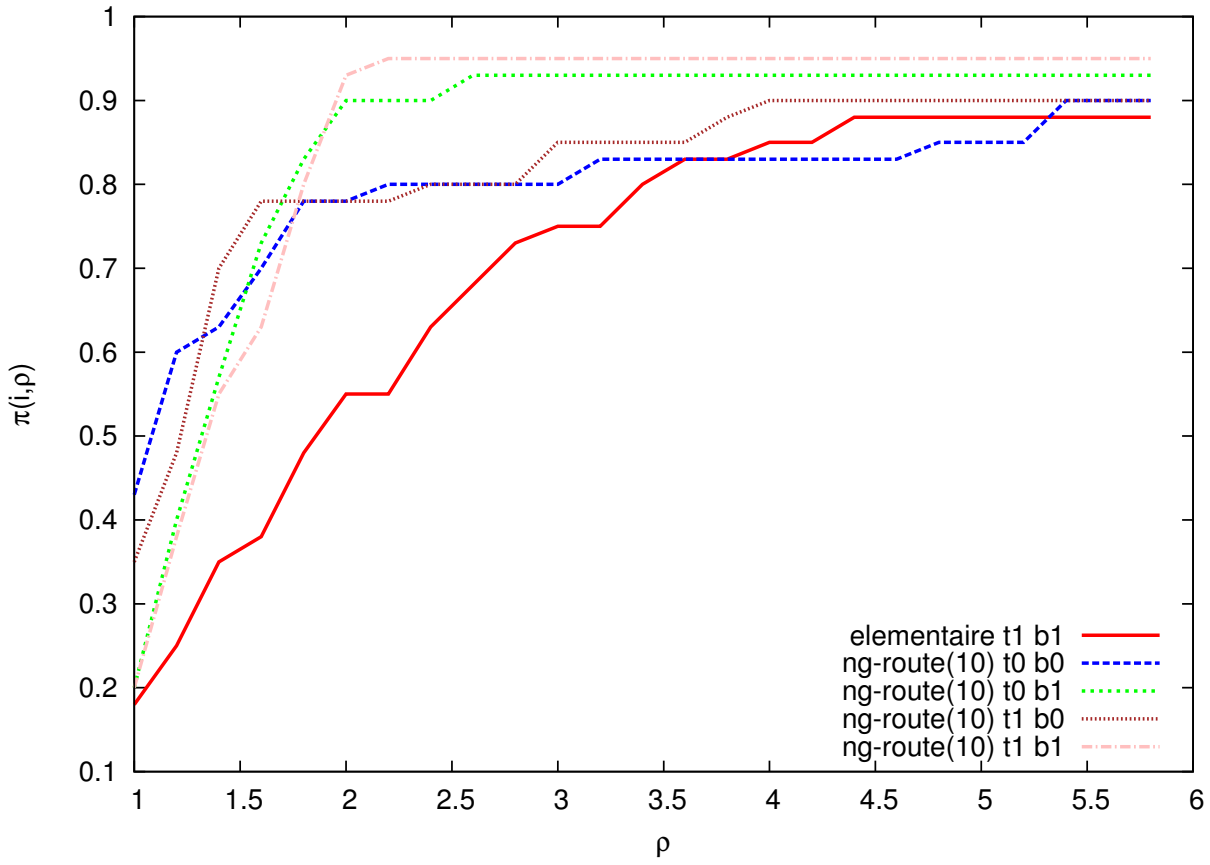

Figure 5.1 Courbes de profil

5.2.3 Comparaison de la méthode ng-route(10) TB avec celle de (Christiansen et Lysgaard, 2007)

Comme en témoigne le tableau 5.5, la méthode ng-route(10) TB est très compétitive par rapport à celle de (Christiansen et Lysgaard, 2007). En effet, toutes les instances sauf A-n33-k5, A-n39-k5 et P-n40-k5 sont résolues plus rapidement avec notre algorithme qu'avec

celui de (Christiansen et Lysgaard, 2007). De plus, ces temps tiennent compte du facteur de conversion de 1,98. Sans cette mise à l'échelle, notre algorithme se révèle plus rapide pour toutes les instances.

Tel qu'attendu, le ratio $\frac{\text{Lb root}}{\text{UB}}$ est plus élevé pour notre algorithme. En effet, le sous-problème avec voisinage que nous utilisons s'assure également d'éliminer les 2-cycles, il procure donc une borne inférieure plus élevée que le sous-problème de (Christiansen et Lysgaard, 2007). Cette augmentation de la borne inférieure contribue souvent à accélérer considérablement la résolution de l'algorithme. Les instance E-n22-k4, E-n33-k4, A-n39-k5 et P-n23-k8 a été résolues directement au noeud racine, alors que les instances P-n40-k5, P-n21-k1 et P-n16-k8 ont nécessité uniquement l'identification d'un ensemble d'inégalités valides.

On remarque également que notre méthode utilise beaucoup moins de noeuds de branchement que (Christiansen et Lysgaard, 2007), car ces derniers n'ont pas ajouté d'inégalités valides. Même en comptabilisant le nombre de fois où l'on a ajouté des inégalités valides, notre processus de branchement nécessite beaucoup moins de noeuds. Par contre, notre algorithme nécessite habituellement plus de résolutions du problème maître. Le tableau 5.5 présente l'information suivante :

- $\frac{\textbf{LB au noeud racine}}{\textbf{UB*}}$: Ratio de la borne inférieure obtenue au nœud racine par rapport à la solution optimale. Cette valeur indique le gap à fermer au nœud racine pour obtenir la solution optimale.
- **Temps** : Temps total nécessaire pour obtenir la solution optimale de l'instance.
- **# Noeuds de branchement** : Nombre total de nœuds de branchement ;
- **# Noeuds Coupes** : Nombre total de fois où des inégalités valides de type CC ou SRI ont été identifiées et ajoutées au problème maître ;
- **# Résolutions du PM** : Nombre de fois que le problème maître a été résolu.
- **Facteur d'accélération** Ratio $\frac{\text{Temps}_{\text{Lysgaard}}}{\text{Temps}_{ng\text{-route}-10}}$. Lors du calcul de ce ratio, on suppose que pour toutes les instances irrésolues par (Christiansen et Lysgaard, 2007), le temps de calcul est de $\frac{1200}{1,98} = 606,06$ secondes. Comme l'instance P-n45-k5 est seulement résolue quelques secondes avant la limite de 1200 secondes, on ne calcule pas ce ratio pour cette dernière. On ne calcule pas non plus ce ratio pour les instances A-n60-k9 et E-n51-k5 que nous ne parvenons pas à résoudre ni pour les instances résolues presque instantanément (près de 0 secondes) par notre algorithme. On utilise $\geq$ pour indiquer que le ratio réel est au mois supérieur à cette valeur.

Tableau 5.5 Résultats finaux et comparaison avec (Christiansen et Lysgaard, 2007)

Instance	ng-route (10), TB					(Christiansen et Lysgaard, 2007)				Facteur d'accélération
	Temps	$\frac{\text{LB root}}{\text{UB}}$	# Noeuds de branchement	# Noeuds Coupes	# Résolutions du PM	Temps	$\frac{\text{LB root}}{\text{UB}}$	# Noeuds de branchement	# Résolutions du PM	
A-n32-k5	24,1	0,98	0	7	90	142,4	0,96	2467	67	5,9
A-n33-k5	5,3	1	0	6	59	4	0,99	117	44	0,8
A-n33-k6	4,9	1	0	4	55	24,7	0,98	909	43	5,1
A-n34-k5	9	0,98	0	6	78	#	0,97	16059	62	$\geq$ 67,3
A-n36-k5	46,9	0,98	0	10	117	#	0,92	8035	113	$\geq$12,9
A-n37-k5	23,4	0,98	0	7	106	#	0,97	9191	95	$\geq$25,9
A-n37-k6	22,8	0,99	2	11	111	#	0,98	16195	67	$\geq$26,6
A-n38-k5	42,8	0,97	6	11	142	#	0,95	14499	55	$\geq$14,2
A-n39-k5	5	1	0	0	44	1,5	1	9	73	0,3
A-n39-k6	19,3	0,99	0	8	102	140,9	0,97	2431	84	7,3
A-n44-k6	125	0,99	12	21	214	#	0,98	11077	59	$\geq$4,9
A-n45-k6	86,3	0,98	2	11	127	#	0,9	9313	95	$\geq$7,0
A-n45-k7	30,5	1	0	9	100	445,5	0,99	5365	84	14,6
A-n46-k7	18,7	0,99	0	4	63	#	0,98	8149	84	$\geq$32,4
A-n48-k7	27,5	0,99	0	6	83	#	0,93	7729	91	$\geq$22,0
A-n53-k7	512,4	0,99	12	23	247	#	0,93	5385	128	$\geq$1,2
A-n54-k7	310,5	0,99	6	18	209	#	0,94	5925	98	$\geq$2,0
A-n55-k9	33,2	0,99	0	7	88	#	0,91	9979	55	$\geq$18,3
A-n60-k9	#	0,42	20	35	355	#	0,93	6889	108	#
E-n22-k4	0,1	1	0	0	19	0,5	1	9	34	5
E-n33-k4	32,6	1	0	0	67	43,4	0,99	73	148	1,3
E-n51-k5	#	0,95	4	29	258	#	0,97	2771	138	#
P-n16-k8	0	1	0	1	12	0	1	17	8	#
P-n19-k2	9,7	0,94	0	7	85	77,3	0,94	1815	63	8,0
P-n20-k2	128,8	0,95	6	15	168	177,8	0,95	3191	88	1,4
P-n21-k2	2,2	1	0	1	44	2,5	0,99	27	100	1,1
P-n22-k2	46,7	0,97	0	8	89	110,6	0,97	1335	103	2,4
P-n22-k8	0	1	0	2	20	0	1	65	18	#
P-n23-k8	0	1	0	0	11	0,5	1	0	19	#
P-n40-k5	6,9	1	0	1	50	4,5	1	35	80	0,7
P-n45-k5	1193,8	0,98	12	34	506	#	0,95	4561	101	#
P-n50-k10	54,6	0,99	16	28	194	#	0,95	18901	50	$\geq$11,1
P-n50-k7	40,4	0,99	0	11	96	#	0,95	5311	67	$\geq$15,0
P-n50-k8	35,7	0,99	2	11	92	#	0,91	10409	55	$\geq$17,0
P-n51-k10	10	0,99	0	8	70	217,2	0,99	5431	57	21,7
P-n55-k10	26,4	0,99	0	10	83	#	0,92	11257	56	$\geq$23,0
P-n55-k15	19,9	1	36	32	208	400	0,99	20027	21	20,1
P-n55-k7	103,8	0,99	0	14	114	#	0,94	2441	102	$\geq$5,8
P-n60-k10	417,5	0,99	34	52	387	#	0,95	4969	59	$\geq$1,5
P-n60-k15	7	1	0	7	50	#	1	19029	29	$\geq$86,6

5.2.4 Analyse des résultats - performance

La bonne performance de notre algorithme peut être expliquée par la nature des instances ainsi que l'interaction de plusieurs facteurs lors de la résolution :

◇ Taux de remplissage optimal des véhicules

Le rapport entre la capacité d'un véhicule et la somme espérée des demandes influence le taux de remplissage optimal des véhicules. Ce ratio joue également un rôle fondamental dans l'efficacité de résolution de notre problème. En effet, le fait que la contrainte de capacité soit restrictive par rapport à la demande des clients visités influence significativement la nature des solutions et leurs coûts. Pour évaluer cette statistique, on peut notamment utiliser le taux de remplissage espéré optimal. Cette valeur est calculée comme suit : $\frac{\sum_{i \in N} E[\xi_i]}{Qk^*}$ où k^* indique le nombre de véhicules utilisés dans la solution optimale du VRPSD. Ce ratio indique la charge espérée d'un véhicule dans la solution optimale(voir le tableau 5.7). Pour la formulation du VRPSD que nous avons adoptée, le nombre de véhicules est une variable alors que les espérances des demandes et la capacité sont des constantes. Un taux de remplissage optimal espéré élevé indique donc qu'il est profitable de charger à haute capacité les véhicules. Peu de véhicules sont donc nécessaires, mais ces derniers visitent beaucoup de clients.

Pour notre problème, on observe que le nombre de véhicules utilisés dans le problème stochastique est toujours plus grand ou égal que le nombre optimal pour le problème déterministe correspondant. En effet, il n'est pas toujours optimal de charger les véhicules à la même capacité que dans le cas déterministe, car ceci est susceptible d'entraîner des dépassements de capacité et des retours au dépôt coûteux pour le problème stochastique. Ceci implique que le taux de remplissage déterministe optimal est une borne supérieure sur le taux de remplissage espéré optimal. Un taux de remplissage optimal espéré élevé nécessite donc un taux de remplissage déterministe élevé.

Afin que le taux de remplissage déterministe soit élevé, il est nécessaire qu'il soit possible de construire une solution composée de peu de routes visitant beaucoup de clients. Ceci nécessite que la somme des demandes soit petite par rapport à la capacité. Le ratio $\frac{\sum_{i \in N} E[\xi_i]}{Q}$ doit donc être faible, car on a toujours besoin d'au moins $\left\lceil \frac{\sum_{i \in N} E[\xi_i]}{Q} \right\rceil$ véhicules. Ainsi, si le taux de remplissage espéré est élevé, on peut conclure que la somme des demandes est forcément petite par rapport à la capacité (condition nécessaire).

Tel que mentionné par (Christiansen et Lysgaard, 2007), lorsque la capacité est peu contraignante par rapport à la somme des demandes, il est possible qu'il existe beaucoup de routes réalisables. Une telle augmentation de l'espace des solutions peut augmenter le nombre de chemins partiels réalisables, donc le nombre d'étiquettes générées à chaque itération du problème de plus court chemin. Ceci est susceptible d'augmenter considérablement le temps de résolution du sous-problème.

L'existence de nombreuses routes de coût réduit négatif pourrait non seulement ralentir l'algorithme d'étiquetage, mais également augmenter la probabilité que la solution optimale soit fractionnaire. Par ailleurs, il est probable qu'il soit plus difficile d'identifier des coupes de capacité violées pour les instances avec une capacité peu contraignante.

◇ Algorithme tabou

Tel que mentionné à la section 4.3, la méthode taboue semble se révéler bénéfique pour accélérer la résolution de l'algorithme d'étiquetage. De plus, celle-ci semble contribuer à l'obtention de bonnes solutions réalisables rapidement. En effet, les méthodes sans tabou doivent souvent effectuer plus de branchements avant d'obtenir une solution entière de coût raisonnable. Toutefois, il n'y a pas de raisons théoriques valides expliquant ce phénomène.

◇ Algorithme d'étiquetage bidirectionnel

Dans la majorité des instances, l'étiquetage bidirectionnel ne se révèle pas plus rapide que la méthode qui ne l'exploite pas. Cependant, les résultats pour les instances P-n45-k5 et P-n20-k2 dans les tableaux 5.1, 5.2 et 5.3 illustrent clairement leur importance. En effet, ces instances demeurent insolubles sans l'algorithme bidirectionnel, et ce peu importe que l'on utilise l'algorithme tabou ou non.

Toutefois, comme on le voit dans le tableau 5.6, les instances peu contraignantes comme E-n33-k4 et E-n22-k4 peuvent quand même être résolues très rapidement. Dans ces deux cas, la résolution est très rapide, car les routes optimales sont toutes générées lors de la résolution du sous-problème au premier nœud de branchement.

◇ Identification d'inégalités valides

Des tests numériques préliminaires ont démontré que les deux types d'inégalités valides

sont essentielles pour résoudre rapidement la grande majorité des instances. L'ajout des coupes permet de résoudre plus d'instances plus rapidement que l'algorithme sans ces fonctionnalités.

Avec la méthode $ng(10)$ TB, la très grande majorité du temps total de résolution est dédié à la résolution du sous-problème. Le sous-problème nécessite effectivement en moyenne 95 % du temps total de résolution. Comme la résolution du sous-problème est très lourde, il est utile de réduire le nombre de nœuds de branchement pour diminuer le temps total de résolution. À ce chapitre, les inégalités valides peuvent se révéler très efficaces, car elles permettent d'accélérer l'obtention d'une solution entières et contribuent donc à réduire le nombre de nœuds de branchement nécessaires pour obtenir la solution entière optimale.

Même si la capacité du véhicule est peu contraignante et que le gap d'intégrité est plus grand que zéro, il est quand même possible que le processus de résolution soit rapide. Ce sera le cas si on parvient à identifier rapidement un bon nombre d'inégalités valides permettant d'obtenir une solution entière. Comme on l'observe dans le tableau 5.6, P-n50-k7 en est un bon exemple. Même si l'instance possède un taux de remplissage espéré optimal de 0,91 et que le saut d'intégrité est 28% au nœud racine, l'instance est résolue en moins de 1 minute. On note également que cette instance est résolue plus rapidement sans algorithme bidirectionnel (voir le tableau 5.3). La rapidité du processus de résolution peut donc être attribuée à l'identification d'inégalités valides.

À l'opposé, l'algorithme ne parvient pas à identifier suffisamment de coupes de capacité violées pour les instances P-n45-k5 et P-n20-k2. En effet, seulement 20% et 14% des nœuds (noeuds de branchement et nombre de fois que des inégalités valides ont été ajoutées) sont des nœuds où l'on utilise ces inégalités valides. On a donc dû principalement effectuer un branchement basé sur les inter-tâches et le flot adjacent à un client pour obtenir une solution entière. Comme ces instances ont nécessité respectivement 128.8 et 1196,3 secondes à résoudre, on en conclut que les coupes de capacité peuvent se révéler importantes pour des instances peu contraignantes dont le saut d'intégrité est relativement élevé.

◇ Règles de branchement efficaces

Lorsqu'on ne parvient pas à identifier beaucoup d'inégalités valides violées, il est crucial d'utiliser des décisions de branchement qui permettront de réduire rapidement le saut d'intégrité afin de limiter le nombre total de nœuds de branchement. Afin de s'assurer d'utiliser de

bonnes décisions, il est parfois nécessaire de considérer plus d'une règle de branchement. Des expériences préliminaires ont démontré qu'en utilisant uniquement la règle de branchement basée sur les inter-tâches, plus de 170 000 routes devaient être générées avant d'obtenir la solution entière optimale pour l'instance P-n20-k2 alors que moins de 7 000 ont été générées pour obtenir la solution optimale avec les deux règles de branchement.

L'évaluation de règles de branchement n'est pas coûteuse comparativement à la résolution du problème maître. Le processus de branchement total nécessite en moyenne uniquement 1% du temps total de résolution. Il est toutefois possible que l'utilisation de plusieurs règles ne soit pas profitable. Par exemple, il serait inutile de considérer deux règles qui élimineraient toujours les mêmes solutions fractionnaires et qui seraient toujours aussi faciles à identifier et appliquer. Idéalement, il serait donc bénéfique d'utiliser des règles facilement évaluables et complémentaires.

Le tableau 5.6 présente les colonnes suivantes :

- **Instance** : nom de l'instance ;
- **Temps** : temps total pour résoudre l'instance avec l'algorithme ng-route(10) TB ;
- **CC** : information relative aux coupes de capacité ;
- **SRI** : information relative aux « subset row inequalities » ;
- **# Noeuds de branchement** : nombre total de noeuds de branchement dans l'arbre de branchement (excluant le noeud racine) ;
- **# Noeuds Total** : nombre total de noeuds de branchement et de fois où l'on a identifié des inégalités valides (# Noeuds (CC +SRI) + # Noeuds de branchement) ;
- **# Noeuds** : nombre total de fois où des inégalités valides de type CC ou SRI ont été ajoutées ;
- **# Coupes Total** : nombre total de coupes de type CC ou SRI identifiées et ajoutées au problème maître au cours de l'algorithme ;
- $\frac{\textbf{\#Noeuds}}{\textbf{\#Noeuds Total}}$: nombre de fois où l'on a identifié des inégalités valides ou effectué une décision de branchement divisé par le nombre total de noeuds de branchement et de fois où l'on a identifié des inégalités valides. ($\frac{\text{\#Noeuds CC}}{\text{\#Noeuds Total}}$, $\frac{\text{\#Noeuds SRI}}{\text{\#Noeuds Total}}$ ou $\frac{\text{\#Noeuds de branchement}}{\text{\#Noeuds Total}}$). Indique l'importance relative de la méthode (inégalités valides ou branchement) par rapport à l'obtention d'une solution entière ;
- $\frac{\textbf{UB root - LB root}}{\textbf{UB root}}$: ratio indiquant la différence entre la meilleure solution entière et la meilleure solution fractionnaire au noeud racine divisé par la valeur de la meilleur solution entière. Un - indique qu'il n'y a pas eu de branchement ou d'identification

d'inégalités valides violées alors que # indique que l'algorithme n'a pas obtenu de solution entière réalisable au noeud racine (autre que la solution artificielle initiale).

Tableau 5.6 Résultats finaux - Inégalités valides et processus de branchement

Instance	Temps	CC			SRI			Branchement		# Noeuds Total	$\frac{\text{UB root-LB root}}{\text{UB root}}$
		# Coupes Total	# Noeuds	$\frac{\#\text{Noeuds}}{\#\text{Noeuds Total}}$	# Coupes Total	# Noeuds	$\frac{\#\text{Noeuds}}{\#\text{Noeuds Total}}$	# Noeuds de branchement	$\frac{\#\text{Noeuds}}{\#\text{Noeuds Total}}$		
A-n32-k5	24,1	6	1	0,14	90	6	0,86	0	0	7	0,03
A-n33-k5	5,3	29	5	0,83	15	1	0,17	0	0	6	0,28
A-n33-k6	4,9	3	2	0,5	20	2	0,5	0	0	4	0,01
A-n34-k5	9	27	3	0,5	28	3	0,5	0	0	6	0,2
A-n36-k5	46,9	30	4	0,4	90	6	0,6	0	0	10	0,33
A-n37-k5	23,4	12	2	0,29	72	5	0,71	0	0	7	0,68
A-n37-k6	22,8	56	6	0,46	62	5	0,38	2	0,15	13	0,25
A-n38-k5	42,8	22	5	0,29	74	6	0,35	6	0,35	17	0,58
A-n39-k5	5	0	0	0	0	0	0	0	0	0	-
A-n39-k6	19,3	28	5	0,62	45	3	0,38	0	0	8	0,44
A-n44-k6	125	21	4	0,12	189	17	0,52	12	0,36	33	0,35
A-n45-k6	86,3	9	3	0,23	120	8	0,62	2	0,15	13	0,36
A-n45-k7	30,5	8	4	0,44	70	5	0,56	0	0	9	0,28
A-n46-k7	18,7	58	2	0,5	30	2	0,5	0	0	4	0,38
A-n48-k7	27,5	42	4	0,67	30	2	0,33	0	0	6	0,4
A-n53-k7	512,4	37	5	0,14	240	18	0,51	12	0,34	35	0,31
A-n54-k7	310,5	55	7	0,29	126	11	0,46	6	0,25	24	0,66
A-n55-k9	33,2	50	4	0,57	45	3	0,43	0	0	7	0,55
A-n60-k9	#	49	12	0,22	219	23	0,42	20	0,36	55	0,23
E-n22-k4	0,1	0	0	0	0	0	0	0	0	0	-
E-n33-k4	32,6	0	0	0	0	0	0	0	0	0	-
E-n51-k5	#	23	9	0,27	212	20	0,61	4	0,12	33	0,05
P-n16-k8	0	1	1	1	0	0	0	0	0	1	#
P-n19-k2	9,7	4	3	0,43	60	4	0,57	0	0	7	0,74
P-n20-k2	128,8	5	3	0,14	130	12	0,57	6	0,29	21	0,13
P-n21-k2	2,2	3	1	1	0	0	0	0	0	1	0
P-n22-k2	46,7	3	2	0,25	90	6	0,75	0	0	8	0,6
P-n22-k8	0	1	1	0,5	5	1	0,5	0	0	2	0,01
P-n23-k8	0	0	0	0	0	0	0	0	0	0	-
P-n40-k5	6,9	4	1	1	0	0	0	0	0	1	0
P-n45-k5	1193,8	22	9	0,2	272	25	0,54	12	0,26	46	0,06
P-n50-k10	54,6	29	12	0,27	119	16	0,36	16	0,36	44	0,42
P-n50-k7	40,4	26	6	0,55	75	5	0,45	0	0	11	0,55
P-n50-k8	35,7	9	4	0,31	98	7	0,54	2	0,15	13	0,49
P-n51-k10	10	21	4	0,5	56	4	0,5	0	0	8	0,01
P-n55-k10	26,4	23	6	0,6	60	4	0,4	0	0	10	0,51
P-n55-k15	19,9	23	15	0,22	61	17	0,25	36	0,53	68	0,47
P-n55-k7	103,8	22	6	0,43	120	8	0,57	0	0	14	0,61
P-n60-k10	417,5	42	21	0,24	273	31	0,36	34	0,4	86	0,57
P-n60-k15	7	37	5	0,71	26	2	0,29	0	0	7	0,4
Moyenne	91,68	21,00	4,68	0,40	80,55	7,20	0,40	4,25	0,10	16,13	0,34

5.2.5 Analyse de la solution stochastique (instances résolues)

La variabilité des demandes des clients peut considérablement modifier la structure intrinsèque du problème ainsi que sa solution optimale. Il est intéressant de quantifier l'importance de considérer cet aléa. Ceci peut être effectué en considérant la valeur de la solution stochastique, c'est-à-dire l'écart entre la valeur de la solution optimale du problème avec recours et la solution que l'on obtiendrait en utilisant la solution optimale lorsque l'on suppose que les demandes prennent toutes les valeurs espérées (la solution avec valeur moyenne - EV). Cette information est détaillée dans le tableau 5.7. Les colonnes présentent les informations suivantes :

- **Nombre optimal espéré de véhicules** indique le nombre de routes utilisées dans la solution optimale. Tel que mentionné par (Christiansen et Lysgaard, 2007), la considération de la variabilité peut augmenter le nombre de véhicules nécessaires par rapport à la solution optimale du problème déterministe correspondant.

  Même si le nombre de véhicules optimal reste le même pour le problème stochastique et déterministe, la valeur et la nature de la solution peut varier de manière radicale. L'instance P-n55-k10 en est un bon exemple. Bien que la solution optimale utilise 10 véhicules dans les deux cas, les routes optimales du problème déterministe ont un coût 7,38 % plus élevé par rapport à la solution stochastique optimale. Les figures 5.3 et 5.2 illustrent bien comment la solution optimale peut changer lorsqu'on ajoute le recours et que l'on considère l'aléa dans les demandes. D'un autre côté, il est également possible que la solution stochastique n'ait presque pas de valeur par rapport à la solution déterministe. Ceci est notamment le cas lorsque les solutions optimales concordent pour les problèmes stochastique et déterministe.

- **Taux de remplissage espéré optimal** (« Expected Optimal Fill Rate ») indique $\frac{\sum_{i \in N} E[\xi_i]}{Qk^*}$ où k^* indique le nombre de véhicules utilisés dans la solution optimale du VRPSD.

- **Taux de remplissage déterministe optimal** (« Deterministic Optimal Fill Rate ») indique $\frac{\sum_{i \in N} E[\xi_i]}{Qk'^*}$ où k'^* indique le nombre de véhicules utilisés dans la solution optimale du problème déterministe. Comme on utilise au moins autant de véhicules dans la solution du problème avec recours, la valeur du taux de remplissage espéré déterministe est toujours plus grande ou égale au taux de remplissage espéré optimal (stochastique). Ces deux ratios diffèrent précisément lorsque la solution stochastique nécessite plus de

véhicules que la solution déterministe.

- **EEV** indique le coût de déplacement total espéré minimal en utilisant la solution avec valeur moyenne. Cette valeur est obtenue en calculant le coût total espéré avec échec minimal des routes de la solution optimale du VRP déterministe équivalent. En effet, comme on suppose initialement que l'espérance des demandes des clients est donnée par les demandes dans le VRP déterministe, la solution optimale du VRP déterministe correspond à la solution EV (la solution avec valeur moyenne). Tel qu'expliqué brièvement dans la section 3.3, il est important de noter que l'orientation des routes est importante pour le calcul de la solution stochastique optimale. Or, l'orientation des routes n'est pas importante pour le VRP. Par exemple, la route 1,2,3 a exactement le même coût que la route 3,2,1. Cependant, l'ordre de visite des clients joue un rôle important dans la demande cumulée à un noeud donné et donc la probabilité d'échec à ce dernier. Ceci influence le coût de la solution dans le graphe $\mathcal{GS}$. Afin de déterminer la vraie EEV, pour chaque route $p = (i_1, i_2, ..., i_{|p|})$ utilisée dans la solution optimale du VRP correspondant, il faut donc prendre le minimum du coût espéré total de p et $p' = (i_{|p|}, ...i_2, i_1)$.

- **RP(UB*)** correspond à la valeur optimale du VRPSD. Il s'agit également de la valeur du problème avec recours (RP). Tel que mentionné dans la section 3.1, l'inégalité $RP \leq EEV$ est toujours respectée.

- **EEV-RP (VSS)** indique la valeur de la solution stochastique (on a $EEV - RP = VSS$). Cette valeur indique l'importance de considérer l'aléa dans la formulation du VRPSD par rapport au coût total de la solution obtenue.

- $\frac{\textbf{VSS}\times 100}{\textbf{UB*}}$ indique le pourcentage de la VSS par rapport à la solution RP (UB*) (la solution optimale du VRPSD tel qu'indiqué dans les tableaux précédents). Comme on le voit avec les instances A-n39-k6 et P-n55-k15, ce ratio peut atteindre près de 10% de la valeur optimale. Ceci justifie l'utilisation d'un modèle stochastique à deux stages dans un cadre réel. Ce gain devient particulièrement important lorsque le coût total des routes est élevé et que les instances sont suffisamment petites pour être solubles.

- ***Note*** Afin de conserver des résultats cohérents avec les solutions optimales déterministes présentées dans (Baldacci *et al.*, 2007) et (Lysgaard *et al.*, 2004) ainsi que la EEV utiliseé dans (Christiansen et Lysgaard, 2007), le tableau suivant suppose que tous les VRP déterministes avec demandes moyennes ($EV = \arg\min_x\{z(x, \bar{\xi}\})$ corres-

pondants sont résolus en résolvant DEP où le coût d'échec est nul pour tous les arcs et en remplaçant la contrainte d'inégalité sur le nombre de véhicules (4.100) par une égalité. Ainsi, pour chaque instance X-nxx-kyy, on utilise exactement yy véhicules. On observe que pour les instances où la solution avec $\geq$ est strictement plus petite que celle avec $=$, imposer une contrainte d'égalité augmente la EEV et donc la VSS, car la valeur de RP reste la même. Ceci semble logique, car une solution déterministe $\lambda^{\geq}$ qui utilise plus de véhicules et qui a un coût de déplacement total plus petit que celui de la solution $\lambda^{=}$ a de bonnes chances d'avoir un plus petit coût d'échec espéré total que $\lambda^{=}$ dans le graphe $\mathcal{GS}$. Ceci est le cas des instances P-n55-k15 et P-n22-k8 qui utilisent respectivement 16 et 9 véhicules dans leur solution optimale lorsqu'on utilise une inégalité sur le nombre de véhicules.

On note que pour bien calculer la EEV, il faut considérer la solution optimale du problème déterministe avec demandes moyennes (EV) où l'on utilise une *inégalité* plutôt qu'une *égalité* sur le nombre de véhicules. Sinon, on ne considère pas les mêmes modèles et il est possible que la VSS soit plus grande qu'elle l'est en réalité. En effet, si la EV est calculée en imposant un nombre exact de véhicules et que le RP est résolu avec une inégalité, alors l'inégalité $EEV \geq RP$ tiendra toujours, mais la valeur EEV pourrait être plus grande qu'elle l'est en réalité (avec une *inégalité*), car le domaine réalisable est plus contraint. C'est notamment ce qui se passe avec l'instance P-n22-k8. Dans ce cas, la VSS est de 26.8 lorsque la EEV est obtenue en considérant une égalité et RP est obtenu avec une inégalité alors qu'en réalité elle est de 0 lorsque les deux modèles sont résolus avec des inégalités.

On observe par ailleurs que si on résout $DEP/DEPs$ en assignant un coût espéré d'échec nul à chaque arc de $\mathcal{GS}$, alors on obtient la valeur optimale du VRP déterministe correspondant. Effectivement, la contrainte $\sum_{i\in\mathcal{N}'}\sum_{j\in\mathcal{N}'}\mathbb{E}_{\xi}\left[\xi_j\right]x_{ij}^{v} \leq Q; \forall v \in \mathcal{V}$ de DEP est « équivalente » à $\sum_{i\in\mathcal{N}'}\sum_{j\in\mathcal{N}'}\xi_j x_{ij}^{v} \leq Q; \forall v \in \mathcal{V}$ du problème déterministe si les demandes ξ sont connues, car on suppose que les instances du VRPSD sont construites en fixant l'espérance de la demande égale à la demande déterministe du VRP correspondant. Comme $\mathcal{Q}(x) \geq 0$ pour toute instance du VRPSD, toute solution d'une instance du VRPSD a un coût plus grand ou égal à l'instance déterministe correspondante. Cependant, ceci n'est pas le cas pour P-n22-k8 si on résout le problème déterministe en utilisant un *égalité* sur le nombre de véhicules. Dans ce cas, le coût espéré total de la solution optimale du VRPSD avec 8 véhicules ou plus est strictement plus petit que le coût de la solution du VRP déterministe lorsqu'on impose 8 véhicules

(592,97 < 603).

Pour que nos résultats soient cohérents avec ceux de (Christiansen et Lysgaard, 2007), le tableau 5.7 présente les instances avec l'espérance des demandes et la capacité du véhicule divisées par le plus grand commun diviseur.

Tableau 5.7 Analyse de la solution stochastique pour les instances résolues

Instance	Nombre optimal espéré de véhicules	Taux de remplissage espéré optimal	Taux de remplissage déterministe optimal	EEV	RP (UB*)	EEV-RP (VSS)	$\frac{\text{VSS} \times 100}{\text{RP (UB*)}}$
A-n32-k5	5	0,82	0,82	890, 13	853, 60	36, 53	4,28
A-n33-k5	5	0,89	0,89	722, 99	704, 20	18, 79	2,67
A-n33-k6	6	0,9	0,9	816, 58	793, 90	22, 68	2,86
A-n34-k5	6	0,77	0,92	839, 95	826, 80	13, 15	1,59
A-n36-k5	5	0,88	0,88	907, 55	858, 70	48, 85	5,69
A-n37-k5	5	0,81	0,81	709, 83	708, 30	1, 53	0,22
A-n37-k6	7	0,81	0,95	1069, 32	1030, 70	38, 62	3,75
A-n38-k5	6	0,8	0,96	831, 99	775, 10	56, 89	7,34
A-n39-k5	6	0,79	0,95	903, 26	869, 10	34, 16	3,93
A-n39-k6	6	0,88	0,88	960, 81	876, 60	84, 21	9,61
A-n44-k6	7	0,81	0,95	1047, 18	1025, 40	21, 78	2,12
A-n45-k6	7	0,85	0,99	1096, 19	1026, 70	69, 49	6,77
A-n45-k7	7	0,91	0,91	1302, 20	1264, 80	37, 40	2,96
A-n46-k7	7	0,86	0,86	1069, 66	1002, 20	67, 46	6,73
A-n48-k7	7	0,89	0,89	1248, 27	1187, 10	61, 17	5,15
A-n53-k7	8	0,83	0,95	1180,1	1124,2	55,9	4,97
A-n54-k7	8	0,84	0,96	1342, 87	1287, 00	55, 87	4,34
A-n55-k9	10	0,84	0,93	1264, 18	1179, 10	85, 08	7,22
E-n22-k4	4	0,94	0,94	411, 73	411, 50	0, 23	0,06
E-n33-k4	4	0,92	0,92	850, 27	850, 20	0, 07	0,01
P-n16-k8	8	0,88	0,88	512, 82	512, 80	0, 02	0,00
P-n19-k2	3	0,65	0,98	229, 68	224, 00	5, 68	2,54
P-n20-k2	2	0,97	0,97	233, 05	233, 00	0, 05	0,02
P-n21-k2	2	0,93	0,93	218, 96	218, 90	0, 06	0,03
P-n22-k2	2	0,96	0,96	231, 26	231, 20	0, 06	0,03
P-n22-k8	9	0,83	0,93	707, 80	681, 00	26, 80	3,94
P-n23-k8	9	0,87	0,98	662, 31	619, 50	42, 81	6,91
P-n40-k5	5	0,88	0,88	475, 45	472, 50	2, 95	0,62
P-n45-k5	5	0,92	0,92	546,05	533,52	12,53	2,29
P-n50-k7	7	0,91	0,91	792, 20	758, 70	33, 50	4,42
P-n50-k8	9	0,88	0,99	606, 41	582, 30	24, 11	4,14
P-n50-k10	11	0,86	0,95	724, 69	669, 20	55, 49	8,29
P-n51-k10	11	0,88	0,97	859, 24	809, 70	49, 54	6,12
P-n55-k7	7	0,88	0,88	616, 44	588, 50	27, 94	4,75
P-n55-k10	10	0,65	0,65	797, 21	742, 40	54, 81	7,38
P-n55-k15	18	0,5	0,6	1191, 34	1068, 00	123, 34	11,55
P-n60-k10	11	0,86	0,95	831, 24	803, 60	27, 64	3,44
P-n60-k15	16	0,89	0,95	1133, 30	1085, 40	47, 90	4,41

5.2.6 Correction des instances

Tel que mentionné précédemment, la bonne méthodologie consiste à utiliser un programme avec une inégalité sur le nombre de véhicules et à ne pas diviser les demandes et la capacité par le PGCD. Le tableau 5.8 présente donc les résultats avec une inégalité pour le nombre de véhicules et les demandes non-divisées par le plus grand commun diviseur. Comme on le remarque, les problèmes déterministes et stochastiques sont très semblables pour P-n22-k8 et E-n22-k4, car la VSS est 0 pour ces 2 instances. Dans ces deux cas, les solutions optimales déterministes et stochastiques concordent. On observe également que la EEV est plus petite que la valeur présentée dans le tableau 5.7 pour toutes les instances.

Tableau 5.8 Analyse de la solution stochastique pour les instances résolues - instances corrigées

Instance	Nombre optimal espéré de véhicules	Taux de remplissage espéré optimal	Taux de remplissage déterministe optimal	EEV	RP (UB*)	EEV-RP (VSS)	$\frac{\text{VSS} \times 100}{\text{RP (UB*)}}$
P-n22-k8	9	0,83	0,83	592, 97	592, 97	0	0
P-n55-k15	18	0,5	0,56	1118, 95	1068	50,95	4,77
E-n22-k4	4	0,94	0,94	377, 10	377, 10	0	0
E-n33-k4	4	0,92	0,92	878, 41	842, 00	45, 41	5,39

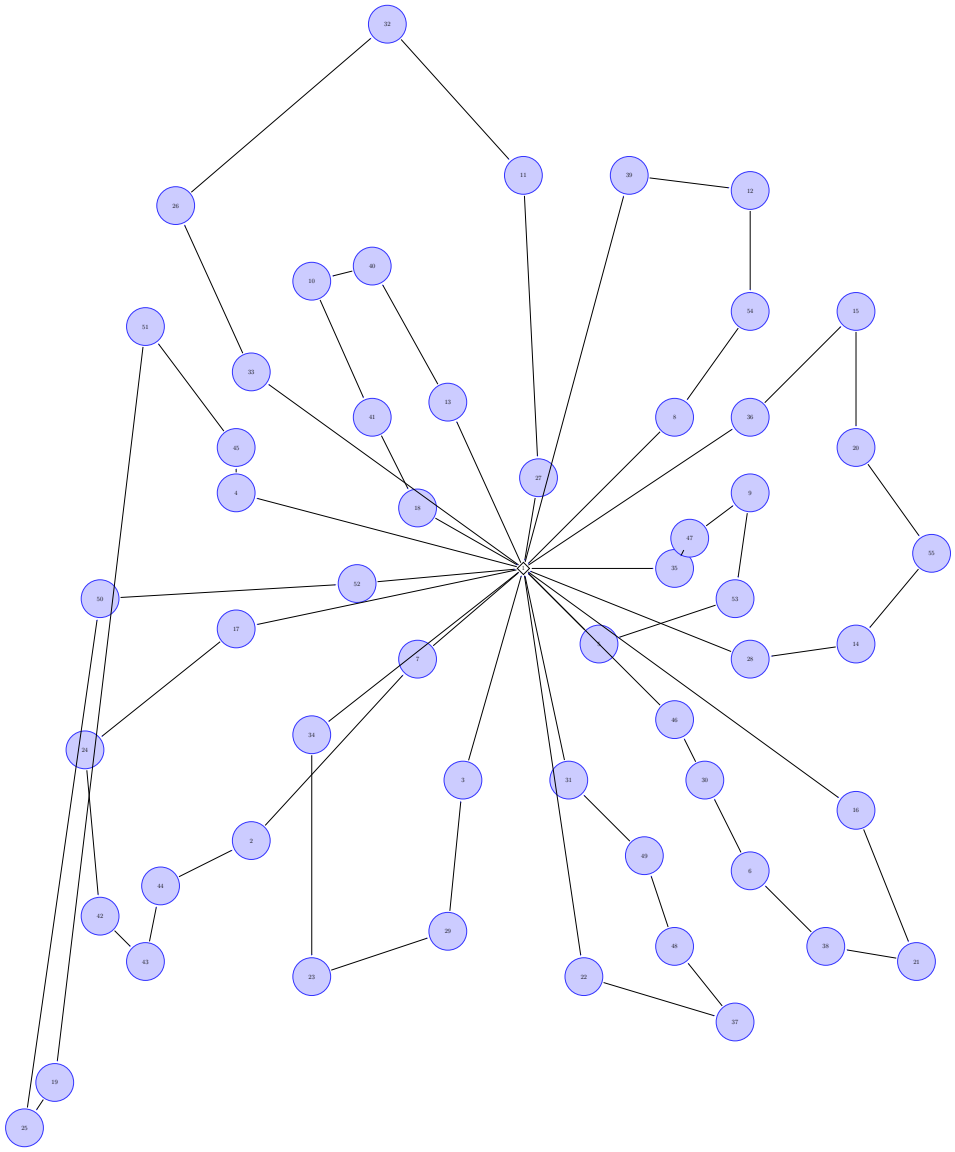

Figure 5.2 Solution optimale pour le problème stochastique pour l'instance P-n55-k10

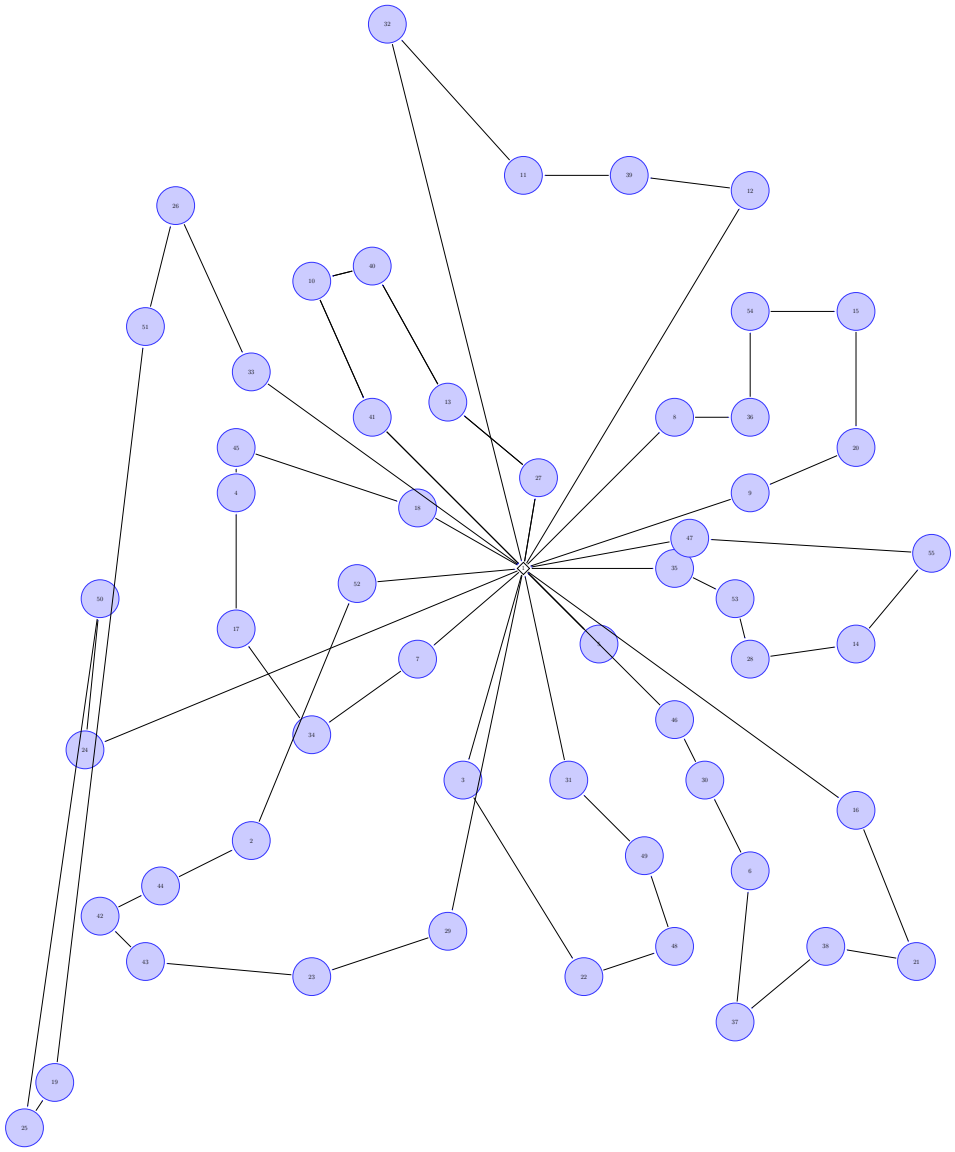

Figure 5.3 Solution optimale pour le problème déterministe pour l'instance P-n55-k10

CHAPITRE 6

CONCLUSION

6.1 Synthèse des travaux

Ce mémoire offre plusieurs contributions théoriques et pratiques importantes.

- Nous avons formalisé le modèle de (Christiansen et Lysgaard, 2007) utilisé pour résoudre le VRPSD en plus de bien le placer dans un cadre de programmation stochastique en deux étapes en nombre entiers. Nous avons également bien expliqué le sens du graphe espace-état $\mathcal{GS}$ ainsi que ses avantages et inconvénients.
- Nous avons bien identifié les distinctions théoriques et algorithmiques entre les différents types de sous-problèmes utilisés lors de la génération de colonnes. Ceci nous a permis d'expliquer nos résultats empiriques et a justifié nos choix d'implémentation.
- Nous avons proposé une règle de dominance agrégée qui accélère la résolution du sous-problème et qui n'a pas été utilisée par (Christiansen et Lysgaard, 2007).
- Nous avons présenté une preuve formelle de la croissance de la fonction EFC(μ,i) par rapport à μ pour tout client $i \in \mathcal{N}$. Cette preuve se révèle essentielle pour démontrer que des routes élémentaires optimales ne peuvent visiter un client plus d'une fois ainsi que pour justifier notre règle de dominance agrégée.
- Nous avons expliqué pourquoi la division des demandes et de la capacité par le plus grand commun diviseur pouvait changer non seulement le coût espéré total des arcs du graphe, mais également modifier significativement la solution optimale.
- Nous avons corrigé l'analyse de la valeur de la solution stochastique effectuée par (Christiansen et Lysgaard, 2007) en nous nous basant sur un VRP déterministe équivalent à celui utilisé pour le VRPSD correspondant, c.-à-d. avec une inégalité sur le nombre de véhicules utilisés.
- Grâce à nos résultats compétitifs, nous avons démontré que les algorithmes de BCP pouvaient se révéler efficaces pour résoudre le VRPSD. En effet, nous parvenons à résoudre à l'optimalité 38 des 40 instances testées en moins de 20 minutes alors que la méthode de (Christiansen et Lysgaard, 2007) parvenait uniquement à en résoudre 18.
- Nous avons effectué des expériences avec un banc de paramètres très riche. Ceci nous a permis d'identifier l'importance des interactions entre les différents modules et concepts. Ces tests nous ont également permis d'établir des pistes de recherche additionnelles.

Un des constats les plus marquants de ce mémoire est l'importance de l'interdépendance entre les différents concepts utilisés par les algorithmes de BCP. Par exemple, plusieurs règles de branchement ainsi que l'algorithme d'étiquetage « arrière » deviennent inutilisables lorsqu'on considère la règle de dominance agrégée. Lorsqu'on ajoute des inégalités valides comme les SRI, on doit également modifier la structure du sous-problème ainsi que toutes les heuristiques et méthodes servant à identifier des colonnes de coût réduit négatif. Par ailleurs, l'utilisation du graphe espace-état $\mathcal{GS}$ peut considérablement ralentir l'évaluation du coût de nouvelles solutions, ce qui peut se révéler désavantageux pour des heuristiques. Finalement, la structure du graphe sous-jacent ainsi que les propriétés comme l'acyclicité jouent un rôle primordial dans la majeure partie des algorithmes. À cause de cette forte interdépendance et de cette sensibilité aux particularités des instances et de leur modélisation, il est difficile de concevoir des algorithmes de BCP réellement robustes et flexibles pouvant être utilisés dans un cadre commercial sans spécialisation extensive.

Les résultats numériques démontrent également l'importance de combiner plusieurs méthodes afin de s'attaquer efficacement aux difficultés spécifiques des différentes instances. Bien que l'ajout de plusieurs méthodes puisse permettre de résoudre des instances insolubles avec des algorithmes plus simples, la combinaison de plusieurs techniques est également susceptible de ralentir la résolution de problèmes facilement solubles avec des méthodes simples. Il peut sembler difficile d'établir un compromis efficace entre spécificité et flexibilité.

Ces résultats sont directement applicables au VRPSD, mais ils se révèlent également importants pour les autres problèmes de tournées de véhicule. Plusieurs conclusions tirées sont aussi applicables aux problèmes pouvant être résolus par BCP comme les problèmes de confection d'horaires d'équipage dans le domaine aérien.

6.2 Limitations de la solution proposée

Bien que nos résultats empiriques démontrent la compétitivité de notre algorithme, les instances E-n51-k5 et A-n60-k9 demeurent irrésolues. Comme notre méthode combine plusieurs techniques à la fine pointe de l'état de l'art et comme nous disposons d'un logiciel de qualité commerciale ainsi que d'ordinateurs performants, il semble difficile d'améliorer facilement notre implantation actuelle. Notre algorithme ne parait donc pas en mesure de résoudre ces instances ; encore moins celles avec une centaine de clients ou plus. La section suivante présente toutefois des améliorations futures envisageables qui permettraient peut-

être de résoudre les instances plus difficiles.

Cette difficulté est bien cernée par (Fukasawa *et al.*, 2006). Ces auteurs affirment que certains algorithmes de résolution de VRP souffrent de rendement marginaux décroissants. Des efforts considérables sont dédiés à la théorie et à l'implantation de meilleurs algorithmes, mais ces derniers ne permettent pas toujours d'accomplir des avancées importantes par rapport à la résolution de nouvelles instances ou l'amélioration du temps de calcul. Il est probable que cette difficulté est reliée à la complexité inhérente de ces problèmes. Afin de réaliser des avancées importantes, il serait donc nécessaire d'utiliser un cadre de résolution différent et non une combinaison de plusieurs méthodes, même si ces dernières sont efficaces.

Il est également important de noter que les caractéristiques particulières des instances rendent difficile l'identification d'une méthode ou d'un banc de paramètres qui dominerait tous les autres. Ce constat est cohérent avec certains travaux récents. Dans leur article, (Fukasawa *et al.*, 2006) déterminent au nœud racine de façon dynamique selon l'instance le type d'algorithme à utiliser. Leurs résultats démontrent effectivement que des algorithmes de BC sont plus efficaces pour résoudre certaines instances alors que dans d'autres cas, les algorithmes de BCP se révèlent plus performants.

6.3 Travaux futurs

Tel que mentionné précédemment, la plus grande difficulté du problème réside dans la résolution du sous-problème et l'identification de routes de coût réduit négatif. Afin d'améliorer notre algorithme, il serait donc utile de se concentrer sur ce point. Il serait intéressant d'implémenter et d'évaluer les améliorations proposées à la section 4.4.3. De plus, il serait possible d'expérimenter avec des métaheuristiques plus poussées comme la recherche à voisinage variable et les méthodes génétiques. Cette dernière technique semble particulièrement prometteuse, car elle offre les résultats les plus performants pour le VRP, est relativement simple et serait facile à adapter au VRPSD (Vidal *et al.*, 2011).

Bien que l'ajout d'une dimension stochastique aux problèmes de tournées de véhicules ne soit pas récente, des travaux plus poussés pourraient être effectués afin d'incorporer une plus grande considération de l'aléa dans la résolution de plusieurs variantes du VRP. Cet ajout est important, car il contribue à augmenter significativement la richesse de ces problèmes et les rapproche de la réalité. La comparaison des solutions optimales stochastiques et déterministes permet également de mieux comprendre leur structure et constitue une analyse de

sensibilité très importante dans un cadre décisionnel réel. Il serait notamment intéressant de considérer des fenêtres de temps aléatoires pour le VRPTW. Dans le but de compléter le cadre de programmation stochastique utilisé dans ce mémoire, il serait aussi utile d'évaluer un modèle de « chance-contrained programming » où l'on optimise le problème en imposant une contrainte sur la probabilité d'échec des routes.

RÉFÉRENCES

BALDACCI, R., BARTOLINI, E., MINGOZZI, A. et ROBERTI, R. (2010). An exact framework for a broad class of vehicle routing problems. *Computer Management Science*, 7, 229–268.

BALDACCI, R., CHRISTOFIDES, N. et MINGOZZI, A. (2007). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115, 351–385.

BALDACCI, R., MINGOZZI, A. et ROBERTI, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *INFORMS, Operations Research*, 59, 1269–1283.

BARD, J., KONTORAVDIS, G. et YU, G. (2002). A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36, 250–269.

BEASLEY, J. E. et CHRISTOFIDES, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19, 379–394.

BERTSEKAS, D. (2005). *Dynammic Programing and Optimal Control - Volume 1*. Athena Scientific.

BERTSIMAS, D. (1992). A vehicle routing problem with stochastic demand. *Operations Reaserch*, 40, 574–585.

BIRGE, J. R. et LOUVEAUX, F. (1997). *Introduction to Stochastic Programming*. Springer.

CHRISTIANSEN, H. C. et LYSGAARD, J. (2007). A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 37, 773–781.

CLARKE, G. et WRIGHT, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, 568–581.

CORDEAU, J.-F., DESAULNIERS, G., DESROSIERS, J., SOLOMON, M. et SOUMIS, F. (2002). *The vehicle routing problem*, SIAM, Monographs on Discrete Mathematics and Applications, chapitre VRP with Time Windows. 157–193.

CORDEAU, J.-F., LAPORTE, G., SAVELSBERGH, M. W. P. et VIGO, D. (2007). *Handbooks in Operations Research and Management Science, volume 14*, Elsevier, North-Holland, chapitre Vehicle Routing. 367–428.

CORMEN, T. H., LEISERSON, C., RONALD, R. et CLIFFORD, S. (2001). *Introduction to Algorithms (2nd ed.)*. MIT Press and McGraw-Hill.

DANTZIG, G. B. (1955). Linear programming under uncertainty. *Management Science*, 1, 197–206.

DESAULNIERS, G., DESROSIERS, J. et SPOORENDONK, S. (2010). *Wiley Encyclopedia of Operations Research and Management Science*, John Wiley & Sons, Inc., chapitre The Vehicle routing problem with time windows : state-of-the-art exact solutions methods. 1–8.

DESAULNIERS, G., LESSARD, F. et HADJAR, A. (2008). Tabu search, partial elementarity, and generalized l-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 45, 387–404.

DESROCHERS, M., DESROSIERS, J. et SOLOMON, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40, 342–354.

DESROCHERS, M. et SOUMIS, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26, 191–212.

DESROSIERS, J. et LUBBECKE, M. E. (2005). *Column Generation*, Springer, New York, chapitre A Primer in column generation. 1–29.

DOLAN, E. et MORE, J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, Series A91, 201–213.

DONGARRA, J. J. (2011). Performance of various computers using standard linear equations software, (linpack benchmark report). Rapport technique, University of Tennessee Computer Science.

DROR, M. (1994). Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5), 977–978.

FEILLET, D., DEJAX, P., GENDREAU, M. et GUEGUEN, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints : application to some vehicle routing problems. *Networks*, 44, 216–229.

FUKASAWA, R., LONGO, H., LYSGAARD, J., POGGI DE ARAGAO, M., REIS, M., UCHOA, E. et WERNECK, R. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming Serie A*, 106, 491–511.

GAREY, M. R. et JOHNSON, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

GELINAS, S., DESROCHERS, M., DESROSIERS, J. et SOLOMON, M. (1995). A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61, 91–109.

GENDREAU, M., LAPORTE, G. et SEGUIN, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29, 143–155.

GENDREAU, M., LAPORTE, G. et SEGUIN, R. (1996a). Stochastic vehicle routing. *European Journal of Operational Research*, 88, 3–12.

GENDREAU, M., LAPORTE, G. et SEGUIN, R. (1996b). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Reaserch*, 44, 469–477.

GOLDEN, B. L. et STEWART, W. J. (1983). Stochastic vehicle routing : A comprehensive approach. *European Journal of Operational Research*, 14, 371–385.

HJORRING, C. et HOLT, J. (1999). New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research*, 86, 569–584.

HOUCK, D. J., PICARD, J. C., QUEYRANNE, M. et VEMUGANTI., R. R. (1980). The travelling salesman problem as a constrained shortest path problem : theory and computational experience. *Opsearch*, 17, 93–109.

IRNICH, S. et DESAULNIERS, G. (2005). *Column Generation*, Springer, New York, chapitre Shortest Path Problems with Resource Constraints. 33–65.

IRNICH, S. et VILLENEUVE, D. (2004). The shortest-path problem with resource constraints and k-cycle elimination for k greater or equal to 3. *INFORMS - Journal on Computing*, 18, 391–406.

JAILLET, P. (1985). *Probabilistic traveling salesman problems*. Thèse de doctorat, Massachussets Institude of Technology.

JEPSEN, M., PERTERSEN, B., SPOORENDONK, S. et PISINGER, D. (2008). Subset-row inequalities applied to the vehicule-routing problem with time windows. *Operations Research*, 56, 497–511.

JULA, H., DESSOUKY, M. et IONNOU, P. A. (2006). Truck route planning in nonstationary stochastic networks with time windows at customer locations. *IEEE Transactions of intelligent transport systems*, 7, 51–62.

KALL, P. et WALLACE, S. W. (1994). *Stochastic programming, 2nd edition*. John Wiley & Sons.

LAPORTE, G. et LOUVEAUX, F. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13, 133–142.

LAPORTE, G., LOUVEAUX, F. et VAN HAMME, L. (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50, 415–423.

LAPORTE, G., MERCURE, H. et LOUVEAUX, F. (1992). The vehicle routing problem with stochastic travel times. *Transportation Science*, 26, 161–170.

LARSEN, J. (1999). *Parallelization of the Vehicle Routing Problem with Time Windows*. Thèse de doctorat, DTU Technical University of Denmark.

LYSGAARD., J. (2004). CVRPSEP : A package of separation routines for the capacitated vehicle routing problem. Rapport technique, Aarhus School of Business, University of Aarhus, Department of Business Studies.

LYSGAARD, J., LETCHFORD, A. et EGLESE, R. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100, 423–445.

PESSOA, A., POGGI DE ARAGAO, M. et UCHOA, E. (2008). *The vehicle routing problem ; latest advances and new challenges*, Springer, chapitre Robust Branch-Cut-and-Price Algorithms for Vehicle Routing Problems. 297–325.

RIGHINI, G. et SALANI, M. (2006). Symmetry helps : Bounded bi-directional dynamic programming. *Discrete Optimization*, 3, 255–273.

SOLOMON, M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35, 254–265.

SPLIET, R. et GABOR, A. F. (2012). The time window assignment vehicle routing problem. Rapport technique, Econometric institue, Erasmus University.

TILLMAN, F. (1969). The multiple terminal delivery problem with probabilitic demands. *Transportation Science*, 3, 192–204.

VIDAL, T., CRAINIC, T. G., GENDREAU, M., LAHRICHI, N. et REI, W. (2011). A hybrid genetic algorithm for the multi-depot and periodic vehicle routing problems. Cahier de notes du CIRRELT (Centre interuniversitaire sur les reseaux d'entreprise, la logistique et le transport).

ANNEXE A : PREUVE DE LA CROISSANCE MONOTONE DE LA FONCTION EFC (μ_i, i) SELON μ_i, LA DEMANDE CUMULÉE À UN CLIENT i DONNÉ LORSQUE LA DEMANDE CUMULÉE À UN CLIENT SUIT UNE DISTRIBUTION DE POISSON

On cherche à montrer que le coût total espéré d'échec au client i sachant que la demande cumulée à ce client est de μ ; (EFC(μ, i)) est croissante en μ. C'est-à-dire EFC$(\mu, i) \leq$ EFC$(\mu + 1, i)$ $\forall \mu \in \{0, 1...Q-1\}$. Comme EFC$(\mu, i) : \mathbb{N} \to \mathbb{R}$ et que EFC(μ, i) est dérivable sur $\mathbb{N}$, alors cette propriété est vérifiée si $\frac{d}{d\mu}$EFC$(\mu, i) \geq 0 \, \forall \, \mu \in \{0, 1, ...Q\}, \forall i \in V$.

On se rappelle que UFAIL(μ, i, uQ) définit la probabilité d'obtenir le u^e échec au client i sachant que la demande cumulée à ce client est de μ et $E[\xi_i]$ représente sa demande espérée si les véhicules ont une capacité de Q. On a :

$$\begin{aligned} \text{UFAIL}(\mu, i, uQ) &= P(\Psi(\mu - E[\xi_i]) \leq uQ) - P(\Psi(\mu) \leq uQ) \\ &= \sum_{k=0}^{uQ} P(\Psi(\mu - E[\xi_i]) = k) - P(\Psi(\mu) = k) \end{aligned}$$

où $P(\Psi(\lambda) \leq uQ)$ définit la probabilité qu'une variable de Poisson de paramètre λ prenne une valeur plus petite ou égale à uQ. On se souvient également que FAIL(μ, i) représente le nombre espéré d'échecs au client i sachant que la demande cumulée à ce client est de μ. On trouve :

$$\begin{aligned} \text{FAIL}(\mu, i) &= \sum_{u=1}^{\infty} \text{UFAIL}(\mu, i, uQ) \\ &= \sum_{u=1}^{\infty} \sum_{k=0}^{uQ} P(\Psi(\mu - E[\xi_i]) = k) - P(\Psi(\mu) = k) \\ &= \sum_{u=1}^{\infty} \sum_{k=0}^{uQ} \left(\frac{(\mu - E[\xi_i])^k}{e^{\mu - E[\xi_i]} k!} - \frac{\mu^k}{e^{\mu} k!} \right) \end{aligned}$$

On a :

$$\frac{d}{d\mu}\mathrm{EFC}(\mu,i) = \frac{d}{d\mathrm{FAIL}(\mu,i)}\mathrm{EFC}(\mu,i)\frac{d}{d\mu}\mathrm{FAIL}(\mu,i) \tag{1}$$

$$= 2c_{0i}\frac{d}{d\mu}\sum_{u=1}^{\infty}\sum_{k=0}^{uQ}\left(\frac{(\mu-E[\xi_i])^k}{e^{\mu-E[\xi_i]}k!}-\frac{\mu^k}{e^{\mu}k!}\right) \tag{2}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\sum_{k=0}^{uQ}\left(\frac{d}{d\mu}\frac{(\mu-E[\xi_i])^k}{e^{\mu-E[\xi_i]}k!}-\frac{d}{d\mu}\frac{\mu^k}{e^{\mu}k!}\right) \tag{3}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\sum_{k=0}^{uQ}\left(\frac{k(\mu-E[\xi_i])^{k-1}}{e^{\mu-E[\xi_i]}k!}-\frac{(\mu-E[\xi_i])^k}{e^{\mu-E[\xi_i]}k!}+\frac{\mu^k}{e^{\mu}k!}-\frac{k\mu^{k-1}}{e^{\mu}k!}\right) \tag{4}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\sum_{k=0}^{uQ}\left(\frac{k(\mu-E[\xi_i])^{k-1}}{e^{\mu-E[\xi_i]}k!}-\frac{k\mu^{k-1}}{e^{\mu}k!}\right)-\sum_{k=0}^{uQ}\left(\frac{(\mu-E[\xi_i])^k}{e^{\mu-E[\xi_i]}k!}-\frac{\mu^k}{e^{\mu}k!}\right) \tag{5}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\sum_{k=1}^{uQ}\left(\frac{(\mu-E[\xi_i])^{k-1}}{e^{\mu-E[\xi_i]}(k-1)!}-\frac{\mu^{k-1}}{e^{\mu}(k-1)!}\right)+0-\sum_{k=0}^{uQ}\left(\frac{(\mu-E[\xi_i])^k}{e^{\mu-E[\xi_i]}k!}-\frac{\mu^k}{e^{\mu}k!}\right) \tag{6}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\sum_{j=0}^{uQ-1}\left(\frac{(\mu-E[\xi_i])^j}{e^{\mu-E[\xi_i]}j!}-\frac{\mu^j}{e^{\mu}j!}\right)-\sum_{k=0}^{uQ}\left(\frac{(\mu-E[\xi_i])^k}{e^{\mu-E[\xi_i]}k!}-\frac{\mu^k}{e^{\mu}k!}\right) \tag{7}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\sum_{j=0}^{uQ-1}\left(\frac{(\mu-E[\xi_i])^j}{e^{\mu-E[\xi_i]}j!}-\frac{\mu^j}{e^{\mu}j!}\right)-\sum_{k=0}^{uQ-1}\left(\frac{(\mu-E[\xi_i])^k}{e^{\mu-E[\xi_i]}k!}-\frac{\mu^k}{e^{\mu}k!}\right) \tag{8}$$

$$+\frac{\mu^{uQ}}{e^{\mu}(uQ)!}-\frac{(\mu-E[\xi_i])^{uQ}}{e^{\mu-E[\xi_i]}(uQ)!} \tag{9}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\left(\frac{\mu^{uQ}}{e^{\mu}(uQ)!}-\frac{(\mu-E[\xi_i])^{uQ}}{e^{\mu-E[\xi_i]}(uQ)!}\right) \tag{10}$$

$$= 2c_{0i}\sum_{u=1}^{\infty}\left(P(\Psi(\mu)=uQ)-P(\Psi(\mu-E[\xi_i])=uQ)\right) \tag{11}$$

Si on peut montrer que $P(\Psi(\mu) = uQ) - P(\Psi(\mu - E[\xi_i]) = uQ) \geq 0 \; \forall \mu, i, u$ alors, on aura montré que :

$$\frac{d}{d\mu}\text{EFC}(\mu, i) = 2c_{0i} \sum_{u=1}^{\infty} P(\Psi(\mu - E[\xi_i]) = uQ) - P(\Psi(\mu) = uQ) \tag{12}$$

$$\geq 2c_{0i} \sum_{u=1}^{\infty} 0 \tag{13}$$

$$\geq 0 \; \forall i, \mu \tag{14}$$

On aura ainsi prouvé que la fonction est bien croissante en μ pour tout $E[\xi_i] \leq \mu \leq Q$ et tout $i \in \mathcal{N}$.

On a : $P(\Psi(\mu) = k) - P(\Psi(\mu - E[\xi_i]) = k) \geq 0$ pour $k = uQ$ si et seulement si :

$$P(\Psi(\mu) = uQ) \geq P(\Psi(\mu - E[\xi_i]) = uQ) \Leftrightarrow \frac{\mu^{uQ}}{e^{\mu}} \geq \frac{(\mu - E[\xi_i])^{uQ}}{e^{\mu - E[\xi_i]}} \tag{15}$$

Ceci sera vérifié si :

$$\frac{x^{uQ}}{e^{x}} \geq \frac{(x-1)^{uQ}}{e^{x-1}}$$

pour tout x tel que $1 \leq x \leq Q - 1$; où $x = \mu - E[\xi_i]$. (on se rappelle que $E[\xi_i] \geq 1 \Rightarrow Q - E[\xi_i] \leq Q - 1$ et $Q \geq E[\xi_i] \Rightarrow Q - E[\xi_i] \geq 0$).

La dernière inégalité est vraie si $\frac{x^{uQ}}{e^x}$ est croissante en x pour tout $x: \ 1 \leq x \leq Q-1$:

$$\Leftrightarrow \frac{d}{dx}\frac{x^{uQ}}{e^x} \geq 0 \tag{16}$$

$$\Leftrightarrow \frac{d}{dx}x^{uQ}e^{-x} \geq 0 \tag{17}$$

$$\Leftrightarrow \frac{d}{dx}e^{uQ\ln x - x} \geq 0 \tag{18}$$

$$\Leftrightarrow (\frac{uQ}{x} - 1)e^{uQ\ln x - x} \geq 0 \tag{19}$$

$$\Leftrightarrow \frac{uQ}{x} \geq 1 \tag{20}$$

$$\Leftrightarrow uQ \geq x \tag{21}$$

$$\Leftrightarrow uQ \geq \mu - E[\xi_i] \tag{22}$$

Ce qui est toujours vrai, car $1 \leq \mu - E[\xi_i] \leq uQ - 1 \ \forall \mu, i, u$.

Ainsi :

$$P(\Psi(\mu) = uQ) - P(\Psi(\mu - E[\xi_i]) = uQ) \geq 0 \ \forall u, \mu, i \tag{23}$$

$$\Rightarrow 2c_{0i}\sum_{u=1}^{\infty} P(\Psi(\mu) = uQ) - P(\Psi(\mu - E[\xi_i]) = uQ) \geq 0 \ \forall \mu, i \tag{24}$$

$$\Rightarrow \mathrm{EFC}(\mu, i) \leq \mathrm{EFC}(\mu + 1, i) \ \forall \mu, i, u \tag{25}$$

□